Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA EN TECNOLOGÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

MODELO DE COMERCIALIZACIÓN Y VERIFICACIÓN SEGURA DE LICENCIAS

Alumno: González Calvo, Jose Luis

Directora: Higuero Aperribay, María Victoria

Curso: 2017-2018

Fecha: Bilbao, a 28 de junio de 2018

Resumen

La digitalización de muchos de los servicios que se comercializan hoy en día en Internet ha permitido que el acceso a dichos contenidos se realice de manera rápida y sencilla. Sin embargo, las compras asociadas a algunos de esos servicios frecuentemente acarrean una serie de problemáticas relacionadas con la seguridad.

El objetivo de este trabajo es la definición y diseño de un modelo de comercialización de licencias *software*, que no solo facilite la compra y distribución a pequeños proveedores, sino que además permita añadir una serie de garantías de seguridad al usuario final durante el proceso de compra. Mediante el uso de diferentes técnicas de seguridad (algoritmos criptográficos, protocolos seguros, autenticación...etc.), se pretende ofrecer una solución que permita comprobar la validez de la licencia a adquirir, sin la necesidad de haber realizado el pago previamente.

Este sistema permite proporcionar al usuario mayores garantías de seguridad en la compra, de forma que si existe algún tipo de engaño o *scam*, pueda cancelar la transacción sin que este se vea perjudicado. Esto garantiza un incremento en la confianza de los usuarios finales en este tipo de servicios, ayudando al despliegue y mayor utilización de los mismos.

Palabras clave: *Seguridad, Modelo, Comercialización, Validez, Garantías, Confianza*

Abstract

The digitization of many of the services that are currently consumed on the Internet has allowed access to these contents to be made easier and faster. However, online shopping always carry a series of security issues that should be considered.

The aim of this project is to introduce a software-license marketing model, which not only facilitates the purchase and distribution to small suppliers, but also adds a series of guarantees to the user during the purchase process. By using some different security techniques (cryptographic algorithms, secured protocols, authentication...etc.), this model allows to check the validity of the license without the need to make the payment previously.

This solution allows the users to have more control over the purchasing process. If there is some kind of deception or scam, they can cancel the transaction without being affected. Final users' confidence will be improved in terms of using these kind of services, helping with their deployment and extended use.

Keywords: *Security, Model, Marketing, Validity, Guarantees, Confidence*

Laburpena

Gaur egun Internet bidez komertzializatzen diren zerbitzuen digitalizazioak eduki horietara heltzea erraztu egin du. Hala ere, zerbitzu horiekin batera doazen erosketek segurtasunarekin lotutako arazoak dakartzate maiz.

Lan honen helburua lizentziak banatzeko software-eredu bat definitzea eta diseinatzea da, hornitzaile txikiei erosketak eta distribuzioa errazteko, eta erabiltzaileari erosketa-prozesuan segurtasun-bermeak eskaintzeko. Segurtasun-teknikak erabiliz (Algoritmo kriptografikoak, protokolo seguruak, autentifikazioa...) lizentziaren baliozkotasuna konprobatu daiteke, ordainketa egin baino lehen.

Sistema honek erabiltzaileari segurtasun-berme gehiago eskaintzen dizkio, iruzur edo *scam* kasu bat egonez gero inolako kalterik gabe transferentzia deuseztatzeko aukera ematen baitu. Sistema honek erabiltzaileek mota honetako zerbitzuetan daukaten konfiantza handitzea bermatzen du, zerbitzu hauen hedapena eta erabilpena erraztuz.

Gako hitzak: *Segurtasuna, Eredu, Merkaturatzea, Baliozkotasuna, Bermea, Konfiantza*

Índice general

Índice de figuras

Índice de tablas

Lista de acrónimos

ACID *Atomicity, Consistency, Isolation, Durability*

AES *Advanced Encryption Standard*

API *Application programming interface*

BASE *Basically Available, Soft state, Eventual consistency*

BSON *Binary JavaScript Object Notation*

CAP *Consistency, Availability, Partition Tolerance*

CA *Certificate Authority*

CBC *Cipher Block Chaining*

CPU *Central Processing Unit*

CSR *Certificate Signing Request*

DB *DataBase*

DMZ *Demilitarized Zone*

DSA *Digital Signature Algorithm*

ECB *Electronic CodeBook Mode*

EHU *Euskal Herriko Unibertsitatea*

FIPS *Federal Information Processing Standards*

GCM *Galois Counter Mode*

HMAC *Hash-based Message Authentication Code*

HSM *Hardware security module*

HTTPS *Hypertext Transfer Protocol Secure*

HTTP *Hypertext Transfer Protocol*

JSON *JavaScript Object Notation*

JWT *JSON Web Token*

KMIP *Key Management Interoperability Protocol*

LAN *Local Area Network*

MEAN *MongoDB, Express, Angular, NodeJS*

NoSQL *Non Structured Query Language*

QoS *Quality of Service*

RAM *Random Access Memory*

RFC *Request for Comments*

RSA *Rivest–Shamir–Adleman*

SAML *Security Assertion Markup Language*

SQL *Structured Query Language*

SSD *Solid State Drive*

SSH *Secure SHell*

SSL *Secure Sockets Layer*

TCP *Transmission Control Protocol*

TDE *Transparent Data Encryption*

TLS *Transport Layer Security*

UDP *User Datagram Protocol*

UPV *Universidad del País Vasco*

URL *Uniform Resource Locator*

VCS *Version Control System*

WBS *Work Breakdown Structure*

XML *eXtensible Markup Language*

XSS *Cross Site Scripting*

1 Introducción al proyecto

El uso de Internet está actualmente muy extendido en cualquier actividad o sector. En la actualidad, gracias a la evolución y mejora constante de la tecnología y de los dispositivos electrónicos, es muy rápido y sencillo poder acceder a servicios *on-line* desde cualquier ordenador o terminal móvil. El aumento en las velocidades de transferencia, así como los mecanismos de QoS y seguridad implementados, permiten disfrutar de todo tipo de contenidos en la red, lo cual hace unos años era impensable.

Uno de los puntos que ha crecido con más fuerza en la última década es el comercio electrónico. El hecho de poder adquirir productos y servicios remotamente resulta muy cómodo y atractivo para los usuarios. Pero las ventajas no solo se limitan a los clientes, sino que los propios *retailers* también salen beneficiados en términos de promoción, visibilidad o costes. La imagen que se presenta a continuación, datada en el Q2 de 2017, muestra el crecimiento del sector del *e-commerce* en España:

FIGURA 1.1: Evolución trimestral del volumen de negocio de *E-Commerce* en España y variación interanual. **Fuente: CNMC**

Sin embargo, hoy en día todavía es común encontrar a personas que son reacias a realizar compras en la red debido a la cantidad de fraudes y problemas de seguridad que se han encontrado en este tipo de comercios a lo largo de los años.

Cierto es que se han ido implementado sistemas de seguridad cada vez más completos, pero la confianza de los usuarios en este tipo de compras nunca va a llegar al nivel de una tienda convencional, donde existe un contacto físico con el *retailer* y donde los procedimientos de devolución en caso de fallo son mucho más sencillos. Si se considera el caso de productos digitales (*software* en general, contenidos multimedia, suscripciones...etc.) la desconfianza se agrava. El hecho de adquirir productos intangibles, los cuales se procesan, distribuyen y se pagan por la red no supone una ventaja en este sentido.

El desarrollo de este proyecto pretende aportar una solución en este sector, concretamente dentro de la compraventa de licencias *software*, de manera que los consumidores tengan una mayor confianza a la hora de realizar sus compras *on-line*. Este documento describe el modelo desarrollado, el cual se basa en el uso de técnicas de seguridad bastante maduras y ampliamente utilizadas (algoritmos criptográficos, SSL/TLS...etc.). Mediante estos mecanismos se pretende ofrecer una solución segura que permita a los potenciales compradores verificar la validez de las licencias sin haber realizado previamente el pago.

Esta solución está principalmente pensada y diseñada para su utilización por parte de pequeños distribuidores, los cuales se dedican a la compra de licencias en gran volumen a las empresas productoras de contenidos digitales (*software*/videojuegos), y que son posteriormente vendidas a un precio más económico a usuarios finales. Sin embargo, no se descarta que parte de esta solución puede ser utilizada en otros ámbitos del comercio de productos digitales en la red.

2 Contexto

La distribución de contenidos digitales ha supuesto toda una revolución en la forma en la que los usuarios pueden acceder a servicios multimedia. Plataformas como *Netflix* o *Spotify*, las cuales hace unos años disponían de una cuota de mercado muy pequeña, actualmente suponen una gran parte del número de transacciones y tráfico global de Internet. Su crecimiento a nivel de usuarios también ha permitido que sus beneficios económicos aumenten de manera considerable. En la imagen que se muestra a continuación se incluye un gráfico con la evolución de estas plataformas en número de suscripciones:

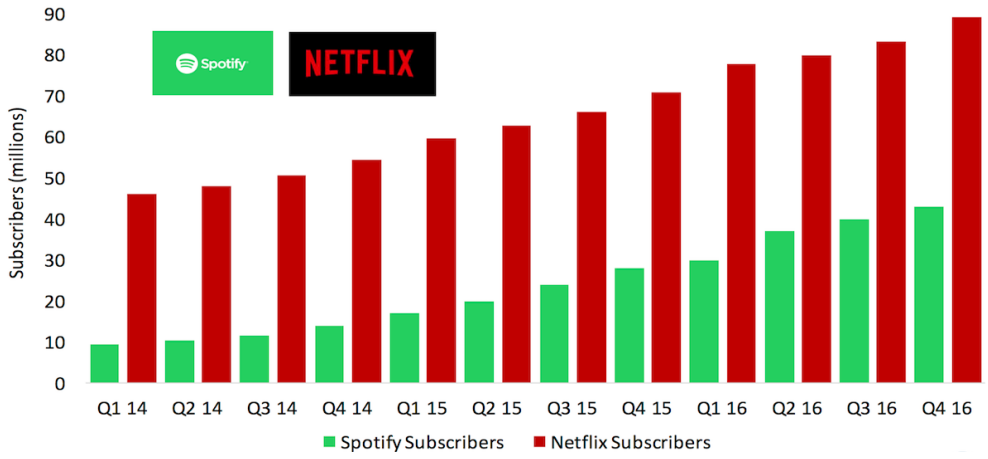


FIGURA 2.1: Evolución en nº de usuarios de las plataformas Netflix y Spotify. **Fuente: MIDIA Research & Company Reports**

Pero esta evolución no solo se limita al caso de contenidos multimedia, sino que también se ha producido un aumento en las ventas de videojuegos y *software* en general. La velocidad de conexión de los usuarios finales se ha visto incrementada de manera considerable en los últimos años, lo cual ha permitido que los proveedores puedan ofrecer este tipo de contenidos a través de Internet. Los costes de distribución se ven muy reducidos y el proceso de obtención del producto es más rápido y cómodo para el usuario.

En la imagen que se muestra a continuación se puede observar como los ingresos obtenidos por las empresas dedicadas al sector de los videojuegos aumentan año tras año, y cómo dicha tendencia parece claro que vaya a mantenerse en años venideros:

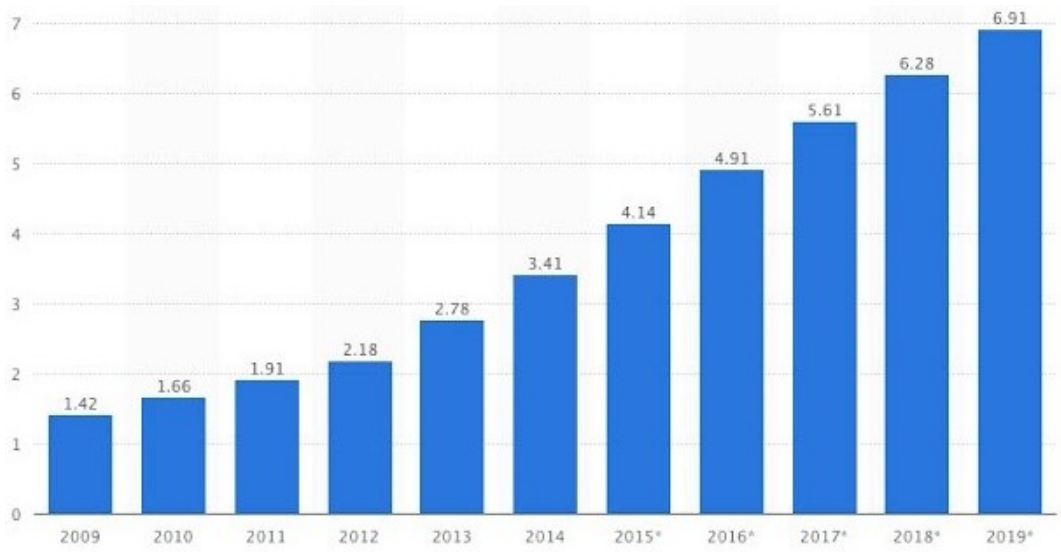


FIGURA 2.2: Ingresos en la industria de los videojuegos en todo el mundo (en billones de dólares).
Fuente: The Competititve Intelligence Unit

Las plataformas líderes actualmente en este sector, como pueden ser *Steam o Battle.net*, han sabido aprovechar de manera muy inteligente estas ventajas para ofrecer un servicio atractivo a los usuarios. Los principales desarrolladores de videojuegos en el mundo establecen alianzas con estas compañías, de forma que estas se dedican a la venta y distribución de dichos productos. Sin embargo, el desarrollo de este mercado va más allá de empresas dominantes que firman acuerdos con productoras de contenidos y distribuyen sus productos.

Actualmente, y desde hace una década aproximadamente, han ido apareciendo numerosas compañías que se dedican a comprar licencias en grandes volúmenes de este tipo de productos para después comercializarlas de forma individual a usuarios finales a un precio más económico. Estos pequeños distribuidores, aunque afirman ser tan seguros y fiables como aquellas grandes empresas que se han citado con anterioridad (*Steam, Battle, Origin*...etc.), han demostrado tener ciertos problemas de seguridad y eficiencia a la hora de gestionar la compra y venta de licencias a usuarios finales:

> [1] *"[...] Marius Mirek, especialista en ventas de G2A, fue parte de un panel en el que presentó la conferencia "G2A Unplugged"donde explicó el modelo de negocios del sitio, el cual cumple una función de intermediario para la compra y venta de claves de acceso (keys) que permiten a los usuarios hacerse de videojuegos por un costo mucho menor al que normalmente tienen en las plataformas en línea. Es a partir de esta situación que algunos miembros de los medios y desarrolladores independientes comenzaron a cuestionar a Mirek sobre el escándalo que rodea a G2A respecto al origen de las claves que ayuda a negociar, el cual es señalado como fraudulento e incluso ha sido vinculado con lavado de dinero. [...]"*

2.1. Modos de operación actualmente considerados

Es necesario conocer más a fondo el funcionamiento de este tipo de comercios para poder desarrollar una solución acorde a las necesidades de ambas partes, tanto clientes como *retailers*. Estas empresas, por norma general, no poseen instalaciones físicas y todos los servicios se gestionan a través de una pagina web o aplicación móvil. En todas ellas se hace uso de certificados de seguridad así como protocolos de comunicación seguros (SSL/TLS).

Como muestra de la forma actual de prestación de este tipo de servicios, se incluyen a continuación las repuestas que se han obtenido al consultar a estas empresas sobre el modo de operación empleado para ofrecer sus servicios:

- Los artículos son comprados por volumen a distintos proveedores y los códigos son escaneados manualmente para posteriormente ser introducidos por empleados dentro de la base de datos de la compañía.
- La empresa comercializa los códigos escaneados previamente, ofreciendo un seguro de compra (el cual es de pago) a los clientes. En caso de entregarse una licencia no válida o ya utilizada, el usuario obtiene más facilidades para obtener una devolución del dinero, aunque esto no es siempre así.
- Como método de pago se acepta el uso de tarjetas de crédito. En el caso de utilizar intermediarios como *PayPal* (el cual proporciona una mejor política de devoluciones), es necesario hacer frente a un pequeño sobrecoste en la compra del producto.

Información obtenida de Instant Gaming Ltd. y G2A Ltd.

Considerando el procedimiento seguido por este tipo de empresas a la hora de realizar la compraventa de licencias, se pretende definir un modelo más eficiente y con mayores garantías de seguridad para el cliente final. El hecho de realizar el escaneo e inserción de licencias de manera manual se presenta como una solución arcaica que puede verse automatizada. Por otro lado, considerando que el precio de las licencias para determinados productos no es muy elevado, en muchas ocasiones no resulta rentable para los usuarios finales tener que pagar un seguro que garantice la validez de la clave a comprar. Por tanto, es necesario generar un modelo seguro que sustituya a los mecanismos de protección utilizados actualmente. Esto no solo beneficiaría al usuario, sino que los *retailers* pueden dar una mejor imagen de garantía y confianza, aumentando así sus ventas.

3 Objetivos y alcance del trabajo

El objetivo principal de este proyecto es definir un modelo para la comercialización segura de licencias *software* que tenga un nivel de eficiencia adecuado y que además proporcione un nivel mayor de garantías al usuario final. Las principales características de la solución deben ser:

- **Seguridad:** Uno de los incentivos que motiva el desarrollo del proyecto es la situación actual en este tipo de mercados. Considerando los intereses de los usuarios finales, se pretende diseñar una solución adecuada que permita garantizar la validez de la licencia a adquirir.
- **Eficiencia:** El modo de operación empleado por las empresas del sector resulta poco eficiente. Se pretende definir una solución que permita automatizar el proceso de compras y gestión de licencias en gran volumen, así como recoger, almacenar y distribuir todos esos datos de forma estructurada.
- **Compromiso:** Es necesario generar confianza dentro de los usuarios para poder asegurar un mayor volumen de ventas. Este compromiso debe surgir dentro de los propios comercios, aportando facilidades y ventajas que ayuden a los consumidores a sentirse más protegidos en sus compras. Este modelo quiere proporcionar una solución con la que las empresas y *e-commerces* puedan ofrecer una mejor imagen de cara al público, depositando total confianza en sus clientes.

Los puntos expuestos a continuación representan otros objetivos adicionales que también resultan de importancia dentro del proyecto:

- **Compatibilidad con múltiples plataformas y tecnologías:** La solución planteada pretende ser definida de forma genérica, de manera que se pueda implementar y funcione sobre cualquier dispositivo o plataforma que se considere. Esto permite llegar a un número mayor de empresas del sector.
- **Modularidad y escalabilidad:** El diseño de la solución esta estructurado en diferentes módulos. Por lo tanto, esto facilita que la implementación de los sistemas pueda realizarse con facilidad, así como añadir, modificar y/o eliminar funcionalidades en función de las necesidades en cada escenario. En términos de escalabilidad ocurre algo similar, al darse la posibilidad de aumentar la capacidad de los equipos que alojan los servicios presentados en el modelo sin la necesidad de modificar la configuración existente.

Este proyecto incluirá el diseño, desarrollo e implementación de la solución que permita alcanzar los objetivos propuestos. El diseño del modelo deberá realizarse acorde con las funcionalidades y características requeridas, analizando las posibles alternativas que pueden llegar a utilizarse. Por otro lado, dicho diseño será complementado con la implementación de un prototipo que incluya los elementos fundamentales que caracterizan al modelo, y que sirva para valorar la viabilidad de la propuesta a diseñar.

4 Beneficios que aporta el trabajo

Tal y como se ha indicado en el apartado anterior, el desarrollo de este proyecto está diseñado con una doble finalidad. No solo se quiere dotar de seguridad y protección a los usuarios cuando realicen sus compras *on-line*, sino que también se quiere fomentar que las empresas distribuidoras puedan funcionar con un nivel de eficiencia adecuado, considerando los modos de operación que se utilizan actualmente (véase apartado **Modos de operación actualmente considerados**).

A continuación se indican los principales beneficios que se derivan de la realización de este trabajo, los cuales se han clasificado de la siguiente forma:

4.1. Beneficios técnicos

- **Alta fiabilidad en la prestación del servicio:** El hecho de estar utilizando tecnologías maduras que son ampliamente utilizadas permite asegurar un buen soporte de las mismas, así como una buen desempeño al aplicarlas dentro del modelo propuesto.
- **Mejora en la gestión y tratamiento de datos:** El modelo planteado contempla el diseño de un proceso automático para la compra y gestión de licencias. Esto permitiría ahorrar tiempo y costes a las empresas distribuidoras, no requiriendo la introducción manual de las estas licencias y realizando el seguimiento de las mismas a través de un servicio web.

4.2. Beneficios económicos

- **Reducción del fraude y aumento de la confianza en las operaciones:** El hecho de proveer un modelo de negocio fiable permite generar un mayor nivel de confianza en los usuarios finales. Esta confianza propicia que los clientes elijan realizar sus compras en este tipo de distribuidores, mejorando así la imagen pública de la empresa y permitiendo obtener un mayor volumen de ventas.
- **Aumento del mercado:** El poder proveer un sistema seguro con el que los clientes se encuentren protegidos ayuda a que las empresas ya existentes se afiancen más en el mercado, y motiva a que dentro del sector de venta electrónica de licencias emerjan más nichos de mercado.

4.3. Beneficios sociales

- **Comodidad para los usuarios finales:** El mayor nivel de despliegue y protección de este tipo de servicios permitirá a un número cada vez mayor de personas adquirir licencias a un precio económico y con las garantías de seguridad adecuadas. Las compras realizadas por estos usuarios podrán realizarse de manera cómoda y asegurando en todo momento las transacciones realizadas con los diferentes elementos implicados en las comunicaciones.

La posterior descripción de la solución definirá como alcanzar cada uno de los beneficios previamente presentados.

5 Análisis de alternativas

A lo largo del desarrollo de este trabajo se han encontrado diferentes elementos en los que se ha planteado la posibilidad de hacer uso de distintas soluciones. Estos elementos han sido sometidos a un proceso de análisis para seleccionar la alternativa que mejor se adapte a las necesidades en cada escenario. Los puntos analizados han sido los siguientes:

- **Securización en el acceso a la base de datos:** Es importante definir un esquema de seguridad efectivo que evite que usuarios no autorizados sean capaces de acceder a los sistemas de almacenamiento disponibles en la topología de red y que, por tanto, accedan a toda la información contenida en ellos.
- **Almacenamiento seguro y estructurado de los datos**: Esta idea va complementada con el punto anterior, ya que pretende definir la forma en la que los datos van a ser estructurados, así como los mecanismos de seguridad a implementar para evitar una posible fuga de datos en caso de que se produzca un acceso no autorizado a los sistemas de almacenamiento.
- **Transferencia segura de información a través de la red:** Finalmente, hay que determinar que mecanismos de seguridad implementar para poder transmitir la información sensible correspondiente a las licencias y los usuarios de forma segura a través de una red pública como es Internet.

FIGURA 5.1: Componentes fundamentales sometidos a estudio

5.1. Securización en el acceso a la base de datos

Las bases de datos son sistemas críticos de almacenamiento de información en las infraestructuras informáticas actuales. Representan un activo muy importante dentro de las empresas e instituciones, por lo que un ataque o vulneración de la seguridad de las mismas puede resultar fatal [2]. En este punto se van a tratar diferentes pautas e ideas que se pueden tener en cuenta para implementar un esquema de seguridad que proteja a las bases de datos de accesos no autorizados. Estos conceptos guardan cierta relación entre ellos, aunque no es necesario la aplicación de todos ellos en conjunto. En función de las necesidades de la organización, se utilizarán unos u otros.

Con el objetivo de determinar cual de las alternativas propuestas resulta ser la más adecuada, se van a definir una serie de criterios previos al desarrollo de cada una de las alternativas. Para la securización del acceso vamos a considerar los siguientes criterios de selección:

- **Costes del equipamiento:** Cada una de las alternativas presentadas a continuación requieren una arquitectura más o menos compleja (servidores, *firewalls*...etc.) [3]. Este punto resulta importante puesto que la selección de una opción u otra puede afectar de manera significativa al coste de la infraestructura de red. Hay que tener en cuenta que muchas empresas no disponen de un presupuesto elevado y esto puede suponer un condicionante para que no adopten esta solución.
- **Nivel de protección ofrecido:** Otro punto a tener en cuenta es de nivel de seguridad que van a aportar cada una de los soluciones descritas en los apartados siguientes. Si la solución propuesta es muy completa y esta bien diseñada, los riesgos de sufrir un ataque y que la información se vea comprometida serán menores [4]. Hay que tener presente que un despliegue más complejo también supone un mayor nivel de dificultad a la hora de realizar la implementación, e igualmente puede suponer un crecimiento en los costes.
- **Rendimiento proporcionado:** La inserción de equipos y mecanismos que proporcionen seguridad en el acceso (*firewalls*, túneles, cifrado...etc.) puede provocar la aparición de *cuellos de botella* dentro de la red interna, especialmente cuando el nivel de tráfico es muy elevado. Por tanto, hay que seleccionar una solución acorde al nivel de calidad de servicio a ofrecer a los clientes.

Una vez indicados los diferentes criterios a tener en cuenta para la securización en el acceso a la base de datos, se procederá a valorar cada una de las alternativas propuestas en torno a sus características principales y su efecto sobre los criterios indicados con anterioridad.

5.1.1. *Hardware* dedicado para el *web-server* y la base de datos

Esta solución está basada en utilizar un mismo equipo físico para alojar ambos servicios. Una de las razones principales por la cual se utiliza este tipo de configuración es la falta de recursos por parte de la empresa. Sin embargo, esto puede acarrear graves problemas de seguridad en caso de que un atacante se haga con el control de uno de los sistemas, por lo que no se recomienda su uso [5]. En resumen, se pueden sacar las siguientes conclusiones de este modelo:

- **Equipamiento requerido:** Esta opción resulta ser la más económica y sencilla de las que se van a plantear puesto que solo se requiere la adquisición o alquiler de un equipo para ofrecer las funcionalidades de *frontend* y *backend*. Sin embargo, en un futuro puede acarrear problemas de escalabilidad.
- **Nivel de protección ofrecido:** La seguridad ofrecida por esta solución resulta muy pobre, puesto que es posible el acceso a este equipo desde el exterior y esto permite que, si un potencial atacante se hace con el control del equipo, este sea capaz de acceder tanto a los elementos del servidor web como a la información de la base de datos.
- **Rendimiento proporcionado:** El hecho de que un mismo equipo tenga que realizar dos funciones simultáneamente supone un riesgo a la hora de ofrecer un servicio óptimo a los clientes. Aunque el equipamiento existente cada vez es más potente, dentro de un ámbito profesional donde se trabaja con una carga de tráfico muy elevada, puede suponer un problema de rendimiento.

5.1.2. Uso de DMZ (*Demilitarized Zone*)

DMZ representa una configuración de red local que esta diseñada para proporcionar seguridad a los equipos de una organización que están expuestos a redes externas no fiables como Internet. De esta forma, un nodo o equipo de la red externa solo puede conectarse a aquellos equipos situados en la DMZ, mientras que el resto de elementos de la red interna se encuentran protegidos por un *firewall* [6].

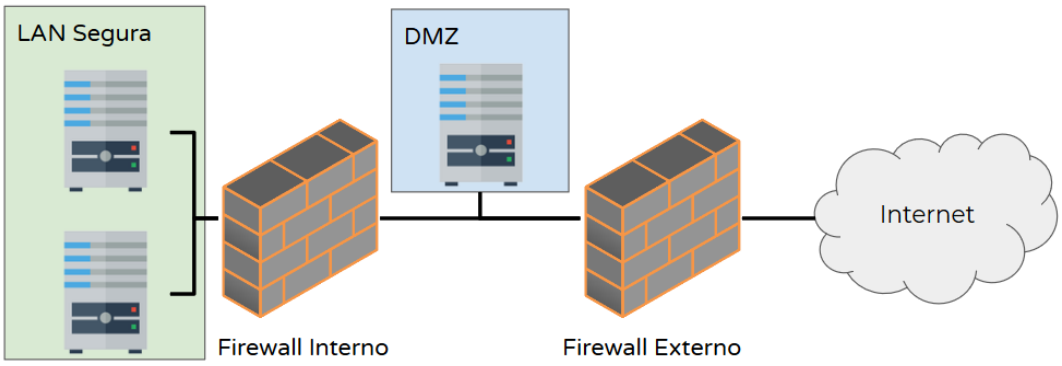


FIGURA 5.2: Arquitectura de red usando DMZ.
Fuente: Angelo State University

A la hora de considerar la implantación de una zona desmilitarizada, se pueden establecer varias topologías. A continuación se presentan las más comunes:

Sistema básico

La primera opción que se puede plantear es situar dentro de la DMZ el servidor web junto con la base de datos, confiando en que el *firewall* sea lo suficientemente fiable como para poder prevenir ataques contra dicha base de datos. Este solo debe permitir el tráfico dirigido hacia aquellos equipos designados para ser "públicos", mientras que el tráfico dirigido a la base de datos es inmediatamente rechazado [5].

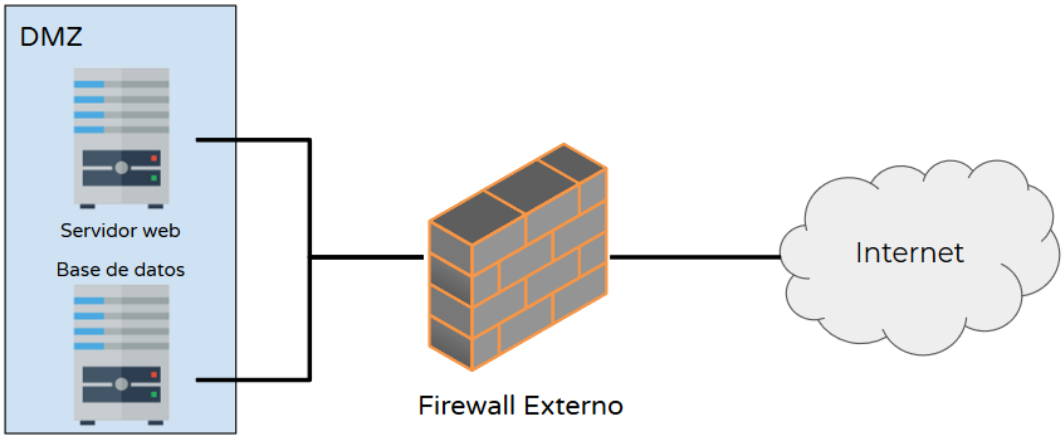


FIGURA 5.3: Sistema básico con DMZ y un *firewall*.
Fuente: Making Your Network Safe for Databases, Duane Winner

Si se consideran las características de este modelo en relación con los criterios definidos anteriormente, se pueden extraer las siguientes conclusiones:

- **Equipamiento requerido:** A partir de este punto hay que destacar la existencia de dos equipos diferenciados para la base de datos y el servidor web. En lo referente al *firewall*, se puede realizar una implementación a nivel *hardware* o *software*.[1] Esto puede suponer un coste superior a la arquitectura presentada en la alternativa anterior, pero en términos generales supone una mejora en seguridad y rendimiento.
- **Nivel de protección ofrecido:** Aunque en principio esta solución parece ideal, puede resultar vulnerable a ataques. El hecho de que el *firewall* bloquee el tráfico dirigido hacia la base de datos no la protege totalmente. En caso de que uno de los elementos situados dentro de la DMZ posea una vulnerabilidad de seguridad y esta sea forzada, el atacante va a ser capaz de usar dicho equipo como lanzadera para atacar otros elementos de la red local, incluida la base de datos. Esta solución resulta poco recomendable, pero es mejor que no tener ningún mecanismo de seguridad implementado.

[1] Dentro de distribuciones Linux existen soluciones *opensource* como *ipchains* o *iptables*. La primera de ellas deja pasar paquetes en función de direcciones IP, puertos o protocolos, mientras que la segunda lleva un control de las sesiones TCP y comunicaciones UDP existentes [7].

- **Rendimiento proporcionado:** El rendimiento va a ser superior en términos generales, puesto que tener dos equipos distintos y que cada uno realice sus funciones permite optimizar el funcionamiento general del sistema. Sin embargo, hay que considerar que puede haber limitaciones, tanto por la existencia de un *firewall* inspeccionando tráfico, como por las comunicaciones a realizar entre los equipos ahora independientes.

Solución basada en 2 *firewalls* y red segura

El situar la base de datos dentro de la DMZ es un problema y se deben plantear otras opciones. La solución que se muestra a continuación proyecta el uso y configuración de un segundo *firewall* para proteger una red nueva, la cual esta separada de la DMZ y es donde se va a situar la base de datos [5]:

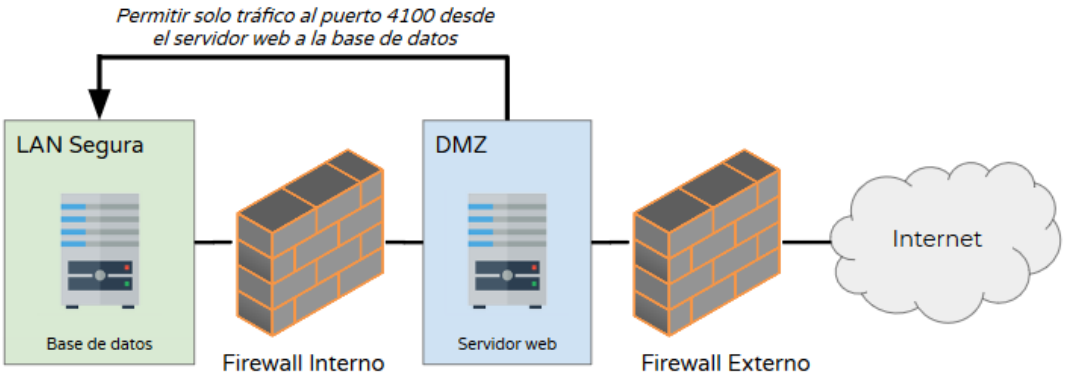


FIGURA 5.4: Uso de 2 *firewalls* para separar el servidor web y la DB.
Fuente: Making Your Network Safe for Databases, Duane Winner

El *firewall* debe ser configurado de forma que se añada un grado más de limitación. Solo permite tráfico muy restringido desde equipos concretos, de forma que, en este caso, el servidor web es el único elemento dentro de la DMZ con permiso para comunicarse con la base de datos, utilizando para ello puertos y conexiones prefijadas [8]. En relación a los criterios utilizados para caracterizar a cada una de las alternativas, se puede concluir lo siguiente:

- **Equipamiento requerido:** La arquitectura vista en este punto no dista mucho de lo presentado anteriormente, excepto por la existencia de un segundo *firewall*. En términos de instalación y mantenimiento esta topología puede llegar a ser muy costosa, por lo que es posible que empresas con un presupuesto limitado no sean capaces de adoptarla.
- **Nivel de protección ofrecido:** Esta solución resulta ser de las más seguras puesto que existe un gran control en las comunicaciones establecidas entre equipos de la DMZ y la red local interna. Una recomendación interesante a tener en cuenta es utilizar diferentes tipos de *firewall* para securizar las diferentes partes de la topología, puesto que si un atacante es capaz de sobrepasar uno de ellos, le puede resultar muy sencillo acceder a la parte correspondiente a la red local segura.

- **Rendimiento proporcionado:** El rendimiento puede verse ligeramente perjudicado sobre la solución anterior puesto que, en este caso, existe un nivel más de protección con la inclusión de un segundo *firewall*. El filtrado de tráfico es realizado en dos puntos de la topología, lo cual puede provocar la aparición de cuellos de botella.

Solución basada en un *firewall* y varios interfaces de red

Tal y como se ha comentado en el apartado anterior, hay diferentes razones que pueden dar pie a que la instalación de un segundo *firewall* sea inconveniente para el sistema (instalación y mantenimiento costosos, configuración de reglas complejas...etc.). En caso de que la instalación de un segundo *firewall* no sea una solución práctica, existe una alternativa basada en el uso de un único *firewall* pero con varias interfaces de red. Esto va a permitir separar la DMZ de la red local interna que se pretende securizar [5]. Un esquema de este modelo puede ser el siguiente:

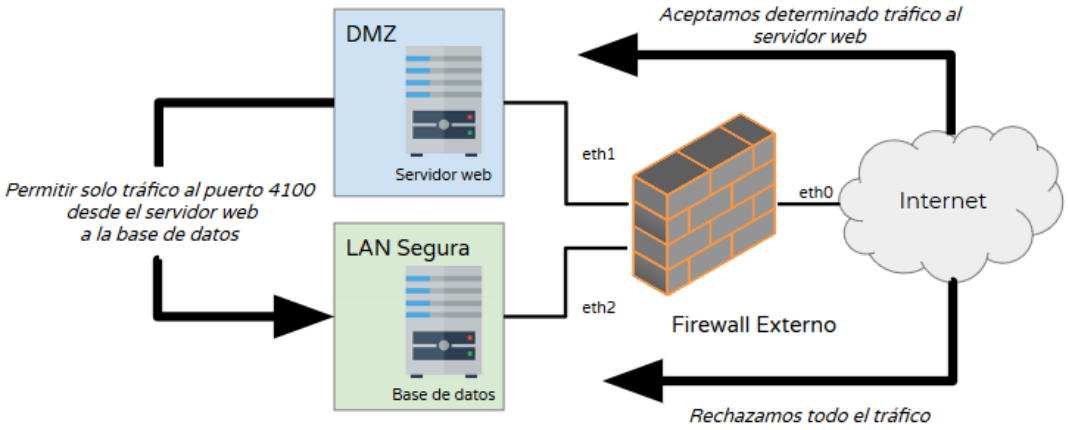


FIGURA 5.5: Uso de un *firewall* y varias interfaces de red.
Fuente: Making Your Network Safe for Databases, Duane Winner

Tal y como se puede apreciar en la imagen, el *firewall* esta configurado al igual que en la solución anterior para limitar el tráfico dirigido a los diferentes equipos dentro de la DMZ. Por otro lado, no se permite la entrada de ningún tipo de tráfico desde la red externa (Internet) hacia la sección local segura. En lo referente al tráfico interno de la organización, hay una excepción dentro de ese mismo *firewall* que permite acceder a la red segura desde la DMZ para unos equipos y puertos/conexiones fijos. En resumen, esta solución posee las siguientes características:

- **Equipamiento requerido:** Una de las ventajas que posee este modelo respecto al presentado en el punto anterior es que requiere unicamente un *firewall* para ofrecer el mismo nivel de seguridad. Sin embargo, el hecho de delegar toda la seguridad en un único *firewall* implica que este debe ser lo suficientemente fiable. Por norma general, esto supone la necesidad de adquirir un *firewall* más completo con un coste más elevado.

- **Nivel de protección ofrecido:** El nivel de seguridad ofrecido por esta solución no dista mucho de la proporcionada por la alternativa anterior. En este caso, tal y como se ha comentado, hay que poseer un *firewall* que sea lo suficientemente fiable y debe estar bien configurado, puesto que toda la seguridad recaerá sobre él.
- **Rendimiento proporcionado:** El rendimiento resulta prácticamente idéntico al presentado por la solución anterior. El hecho de poseer dos equipos ayuda a que cada uno de ellos se encargue de manera óptima de cada tarea, pero la existencia de un *firewall* examinando continuamente el tráfico puede originar cuellos de botella cuando la carga de trabajo es elevada.

5.1.3. Reemplazo de *hubs* por *switches*

La idea planteada en este punto supone un complemento que puede ser incluido dentro de las diferentes arquitecturas que se han presentado a lo largo de la sección. Mediante el uso de un *switch* en vez de un *hub*, se reduce de manera considerable la probabilidad de que los datos transmitidos puedan ser capturados y analizados. Esto es debido a que cualquier equipo que esté conectado a un *hub* es capaz de tener acceso a la información que circula a través del concentrador [9].

El comportamiento de ambos equipos se encuentra representado en la siguiente figura:

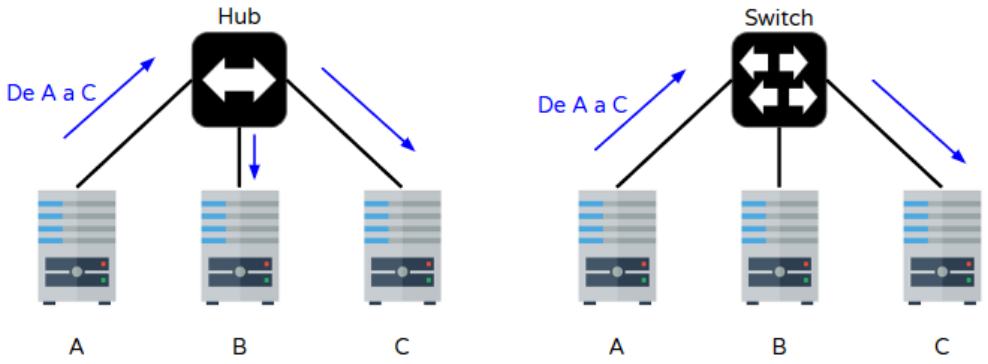


FIGURA 5.6: Diferencias entre *switch* y *hub*.

En caso de que un intruso sea capaz de tomar el control de un equipo en la DMZ, puede configurar la interfaz de red en modo promiscuo[2] y capturar todo el tráfico intercambiado entre el *web-server* y la base de datos. El hecho de reemplazar el *hub* con un *switch* propicia que la comunicación entre dos *hosts* es realizada por medio de un "circuito virtual" que permite que el resto de equipos no vean o detecten el intercambio de tráfico que se produce entre esos dos equipos. Dentro de los criterios de selección se puede tener en cuenta lo siguiente:

[2]**Modo promiscuo**: Configuración de la tarjeta de red que permite al equipo capturar todo el tráfico circulando por la red, sin considerar que sea dirigido hacia él o no.

- **Equipamiento requerido:** Un *switch* supone un coste más elevado que la adquisición de un *hub*.
- **Nivel de protección ofrecido:** Tal y como se ha comentado en la descripción de esta solucion, se puede obtener un mayor nivel de protección al evitar que posibles intrusos hagan uso de *sniffers* para el capturar el tráfico intercambiado entre la DMZ y el área local segura.
- **Rendimiento proporcionado:** Los *switches* disponen por norma general de interfaces más rápidas, además de eliminar dominios de colisión. Por lo tanto, el rendimiento proporcionado resulta óptimo.

5.1.4. Uso de encriptación entre el servidor web y la base de datos

Aún resolviendo la mayoría de problemas de seguridad con los métodos explicados anteriormente, es posible que todavía un atacante sea capaz de acceder a la información compartida entre el *web-server* y la base de datos. Mediante el control de uno de los equipos de la DMZ, podría reconfigurar los *switches* para que todo el tráfico entre el servidor web y la base de datos sea reenviado a otra interfaz desde donde se pueda capturar toda la información [5]. Este ataque se encuentra representado en la siguiente imagen:

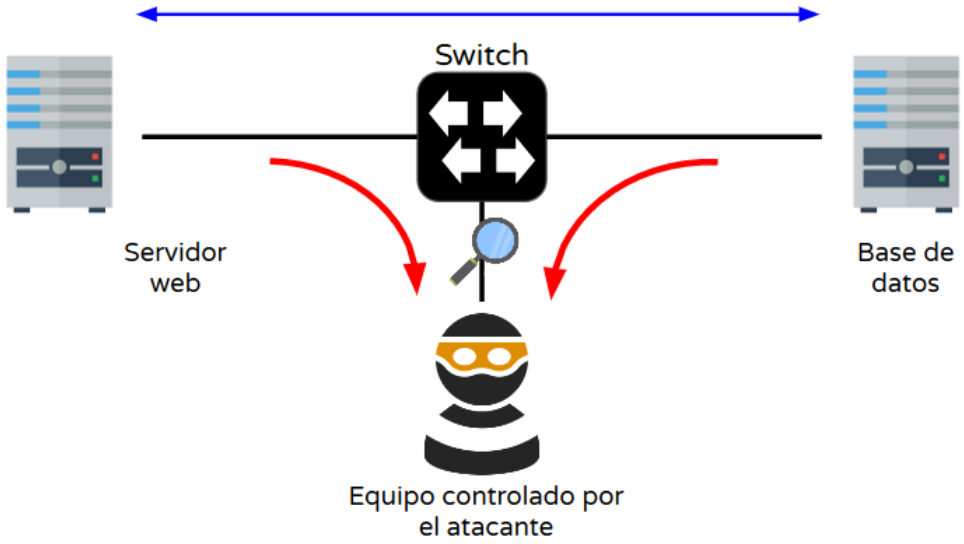


FIGURA 5.7: Reenvío de tráfico a un equipo comprometido

Por lo tanto, la solución final que se plantea es el uso de encriptación en las comunicaciones con la base de datos. Es importante tener en cuenta que la encriptación SSL que se utiliza en la mayoría de navegadores solo codifica la información desde el extremo cliente hasta el servidor web, mientras que las comunicaciones entre el *web-server* y base de datos se realiza como texto plano. Esta solución resulta especialmente necesaria cuando el almacenamiento de los datos se realiza en claro. Sin embargo, en el momento que se opta por un almacenamiento cifrado, esta opción resulta innecesaria. Para integrar la encriptación de los datos entre el servidor web y la base de datos se pueden utilizar diferentes alternativas.

Integración nativa de SSL

En caso de que la base de datos soporte la encriptación nativa de la información, es posible aprovechar esta funcionalidad configurando de la manera que corresponda tanto la propia base de datos como el servidor web. Esto implica, además, que la aplicación web que se esté ejecutando dentro del servidor sea capaz de trabajar también de manera nativa con la encriptación establecida dentro de la base de datos [10]. Las características principales de esta alternativa en relación con los criterios definidos son:

- **Equipamiento requerido:** Implementar SSL de forma nativa en las comunicaciones entre la base de datos y el servidor web implica que ambos equipos sean compatibles con la tecnología mencionada, y en ocasiones las tecnologías utilizadas no tienen estas características incluidas.
- **Nivel de protección ofrecido:** Si bien es cierto que implementar este mecanismo de seguridad garantiza un nivel de protección superior, tal y como se ha mencionado con anterioridad, en caso de que se realice un almacenamiento de los datos cifrado, resulta innecesario.
- **Rendimiento proporcionado:** Implementar mecanismos de cifrado en las comunicaciones siempre tiene una repercusión negativa sobre el rendimiento y el *throughput*.

SSH Port Forwarding

Una alternativa que se plantea en caso de que no sea posible la implementación nativa de SSL es el uso de *SSH Port Forwarding*. Esta tecnología provee un mecanismo para tunelar puertos en las aplicaciones desde el equipo cliente hasta el servidor o viceversa, y es comunmente utilizada para añadir mecanismos de cifrado en aplicaciones que no tienen soporte nativo y/o pasar a través de *firewalls*). En la imagen incluida a continuación se representa el funcionamiento de *SSH Port Forwarding*:

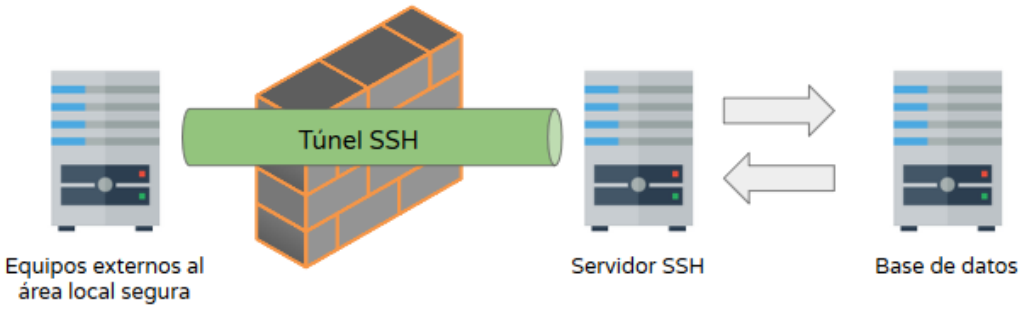


FIGURA 5.8: Funcionamiento de *SSH Port Forwarding*
Fuente: SSH.com

Una ventaja interesante de *SSH Port Forwarding* es que permite ser usado en ambos sentidos. Por lo tanto, se distingue [11]:

- ***Local Forwarding*:** Modo que realiza la configuración del túnel de manera local, mediante la redirección de un puerto local a una dirección/puerto externo.
- ***Remote Forwarding*:** El túnel es iniciado desde el equipo remoto. Esto significa que la base de datos debería realizar la apertura del túnel a la hora de transferir la información. Por tanto, el uso de este método implica que la configuración del *firewall* que protege la zona segura debe filtrar todo el tráfico entrante, asegurando que ningún *host* externo puede iniciar una conexión hacia ningún recurso existente dentro de la red.

En la siguiente imagen se puede apreciar la configuración *Remote SSH Port Forwarding*, con el correspondiente filtrado de tráfico entrante por parte del *firewall*:

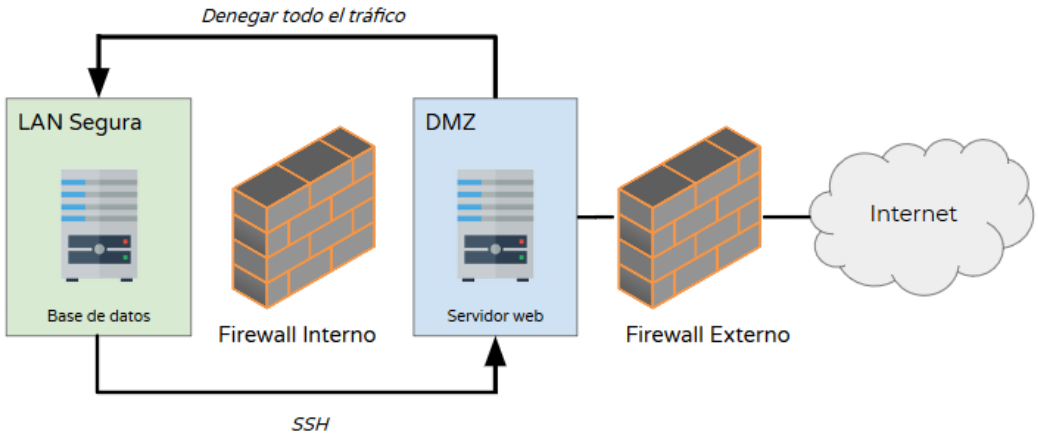


FIGURA 5.9: Uso de *Remote SSH Port Forwarding*
Fuente: Making Your Network Safe for Databases, Duane Winner

Mediante esta configuración, la base de datos solicita al servidor web que establezca un puerto en escucha determinado para su dirección de *loopback*. Todo el tráfico, a través de dicho puerto, va a ser enviado por medio de la sesión SSH hasta la base de datos. En definitiva, de esta solución puede concluirse lo siguiente:

- **Equipamiento requerido:** El sistema propuesto es perfectamente compatible con alternativas de arquitectura descritas anteriormente. La complejidad viene dada por una correcta configuración del *firewall*, así como del servicio de SSH[3].
- **Nivel de protección ofrecido:** Esta alternativa garantiza un nivel de seguridad muy alto, acorde con la complejidad del sistema propuesto.
- **Rendimiento proporcionado:** El uso de varios *firewalls* así como comunicaciones por medio de SSH no garantizan un rendimiento totalmente óptimo en situaciones de mucha carga de tráfico.

[3]SSH no puede ser configurado para ejecutarse automáticamente como un servicio o dominío. Por ello, en muchos casos se requiere una intervención manual para establecer las conexiones

5.2. Almacenamiento seguro y estructurado de datos

En este apartado se van a definir diferentes métodos que pueden ser implementados para el almacenamiento seguro y estructurado de la información. Para ello, se describen alternativas principalmente relacionadas con el tipo de bases de datos que se pueden utilizar para el salvado de los datos, así como distintos métodos de encriptación que pueden ser implementados dentro de la base de datos para evitar un posible filtrado de información en caso de producirse un acceso no autorizado.

5.2.1. Almacenamiento estructurado: Tipos de bases de datos

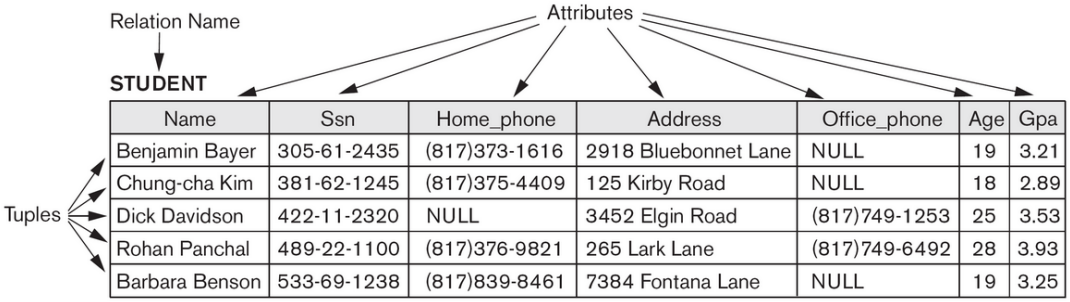
Actualmente existen diferentes clases de bases de datos que pueden utilizarse para el almacenamiento de la información. El uso de un tipo u otro depende principalmente del nivel de tráfico que deben soportar, así como la escalabilidad que ofrecen y las funcionalidades requeridas por el usuario. En este caso, se va a optar por el estudio de la implementación de una base de datos SQL frente una NoSQL debido a que son dos modelos muy utilizados en la actualidad dentro de los escenarios que se plantean en este proyecto [12]. A la hora de realizar la selección de un modelo u otro, se van a tener en cuenta los siguientes criterios de selección:

- **Consistencia de los datos y fiabilidad:** Es importante asegurar que el sistema propuesto es lo suficientemente robusto para gestionar las diferentes transacciones que se soliciten y que en todo momento el estado de la información almacenada sea coherente y estable.
- **Velocidad de procesado y rendimiento:** Este criterio supone un punto importante para asegurar una buena calidad de servicio a los usuarios/clientes. Suponiendo que los dos modelos planteados tienen un comportamiento adecuado cuando la carga de trabajo es pequeña, se va a analizar la capacidad de los dos sistemas para gestionar el tráfico en situaciones con mucha carga de trabajo.
- **Escalabilidad:** Otro punto que resulta importante es la facilidad y disposición de ambos modelos para poder realizar expansiones en la infraestructura, sin que esto implique una modificación en la estructura de datos considerada.

Bases de datos relacionales (SQL)

Una base de datos relacional se define como una base de datos en la que los datos son organizados en base a un modelo relacional. Los datos en este tipo de modelo son representados por medio de tablas denominadas *relaciones*. Cada una de las filas que componen la tabla representan un elemento almacenado, donde, a su vez, cada una de las columnas representan los diferentes atributos que caracterizan al elemento. Esta estructuración puede contemplarse en la siguiente ilustración:

FIGURA 5.10: Criterios de selección sometidos a estudio



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

FIGURA 5.11: Estructuración de datos en un modelo relacional
Fuente: Montana State University

A la hora de realizar la gestión de este tipo de bases de datos se suele utilizar el lenguaje SQL. Mediante este lenguaje basado en consultas, se permiten realizar diferentes tipos de operaciones sobre las bases de datos relacionales, entre ellas, inserción de datos, consultas, actualizaciones/borrado, creación/modificación de esquemas, control de acceso...etc. La mayoría de las bases de datos relacionales garantizan lo que se denominan ***"transacciones ACID"*** [13]. Este término es un acrónimo de las cuatro propiedades que caracterizan este tipo de transacciones:

- **Atomicidad:** Si una operación consiste en una serie de pasos, deben ejecutarse todos, o de lo contrario no se ejecutará ninguno de ellos. Esto asegura que la operación es completada en su totalidad.
- **Consistencia:** Se ejecutarán aquellas operaciones que aseguren que la integridad de los datos es correcta después de realizar la operación. Esto significa que la transacción a ejecutar debe ser válida, respetando la estructura de los datos.
- **Aislamiento:** Asegura que la realización de dos transacciones sobre la misma información sean independientes y no generen ningún tipo de error.
- **Durabilidad:** Una vez completada la transacción, esta no podrá ser deshecha.

El hecho de proporcionar todas las características definidas en este punto supone un sacrificio en el rendimiento aportado por este tipo de modelos. Si estos sistemas

bien ofrecen una capacidad de procesamiento elevada, en caso de tener que trabajar con un volumen de datos excesivo, esta puede verse reducida significativamente [14]. Este comportamiento puede observarse por medio de la ilustración mostrada a continuación:

FIGURA 5.12: Rendimiento de una base de datos relacional en función del volumen de datos. **Fuente: Datajobs**

La escalabilidad en este tipo de bases de datos se suele realizar de forma vertical. Esto significa que una mejora del sistema viene dada por la compra e instalación de equipamiento *hardware* más potente (CPU, RAM, SSD..etc.) [15]. Este método, en principio, resulta sencillo de implementar puesto que unicamente es necesario respaldar los datos y migrarlos al nuevo sistema físico. Sin embargo, podemos encontrarnos con una serie de limitaciones:

- **El coste de instalación puede resultar muy elevado:** La mejoría en el tiempo de respuesta puede ser tan pequeño que no compense la inversión necesaria para adquirir el nuevo equipamiento.
- **El hecho de poseer un mejor equipamiento no asegura una correcta adecuación del flujo de trabajo:** En muchas ocasiones, el problema viene limitado por una definición poco eficiente del esquema funcional de la base de datos. Esto hace que las consultas se ralenticen y el flujo de trabajo no sea óptimo.

Estas limitaciones causan que esta solución no sea suficiente en algunos escenarios, por lo que resulta necesario contemplar el uso de un sistema mixto (escalabilidad vertical y horizontal). El hecho de añadir varios servidores agrupados formando un sistema único ayuda a que se puedan atender a más peticiones simultáneamente, pero también complica la consistencia de los datos al tener que coordinar todos los elementos que conforman el *cluster* [15]. La siguiente imagen muestra la diferencia entre escalamiento vertical y horizontal:

En definitiva, los sistemas relacionales aportan buenas alternativas para aumentar la capacidad del sistema, aunque en la mayoría de situaciones, las limitaciones vienen dadas por las propias funciones de consistencia definidas en este tipo de modelos [16]. De forma resumida, se pueden concluir los siguientes aspectos sobre los modelos relacionales en consonancia con los criterios descritos previamente:

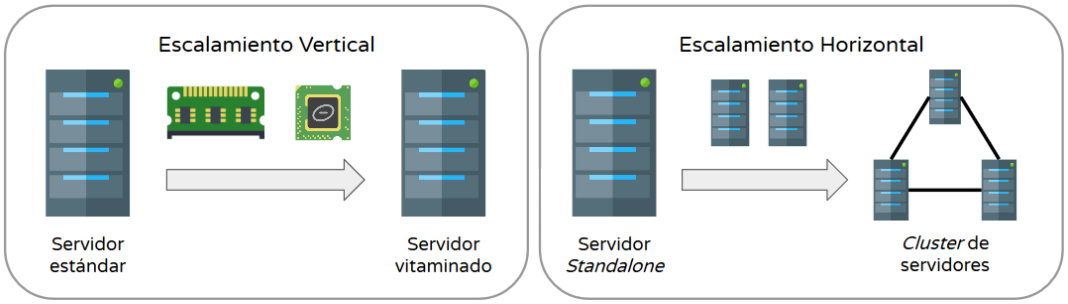


FIGURA 5.13: Escalabilidad vertical vs Escalabilidad Horizontal

- **Velocidad de procesado y rendimiento:** Tal y como se ha indicado a lo largo del desarrollo, el rendimiento de los modelos relacionales es bueno en términos generales. Sin embargo, en caso de tener que gestionar un volumen de datos muy elevado, la capacidad de procesamiento puede verse reducida significativamente (véase Figura 5.12).
- **Escalabilidad:** La escalabilidad en este tipo de modelos se suele realizar de forma vertical, lo que implica que el equipo sea más potente y sea capaz de gestionar el tráfico de manera más rápida. Sin embargo, en algunos casos, la limitación no viene dada por la capacidad de procesamiento, sino más bien por las propiedades inherentes a las transacciones realizadas en este tipo de modelos. Por tanto, es necesario definir la causa que provoca realmente el cuello de botella en el sistema, con el objetivo de no realizar inversiones costosas que posteriormente no vayan a tener ningún beneficio sobre el rendimiento.
- **Consistencia de los datos y fiabilidad:** Los modelos relacionales se caracterizan por disponer de un gran control sobre las transacciones realizadas. Las propiedades *ACID* que se han mencionado con anterioridad aseguran que los datos almacenados siempre se van a encontrar en un estado estable y que las operaciones se ejecuten de forma completa.

Bases de datos no relacionales (NoSQL)

Una base de datos no relacional hace uso de un modelo de gestión de datos donde se prescinde de una estructura concreta, sin considerar tablas previamente definidas. Esto permite obtener unos niveles de flexibilidad y rapidez en las operaciones mucho mas elevados que con base de datos tradicionales [17]. En la imagen mostrada a continuación se indica una posible clasificación de las bases NoSQL:

La clasificación de los sistemas NoSQL puede realizarse en función a diferentes factores. Tal y como se ha mostrado en la figura anterior, una forma de realizar la clasificación de estas bases de datos es considerando el modelo de datos utilizado por el sistema [18]. Los 4 tipos principales considerados son los siguientes:

- **NoSQL orientada a documentos:** Gestionan datos semiestructurados, es decir, documentos. Estos datos son almacenados en un formato estándar como puede ser XML, JSON o BSON. Se pueden utilizar en gran cantidad de proyectos,

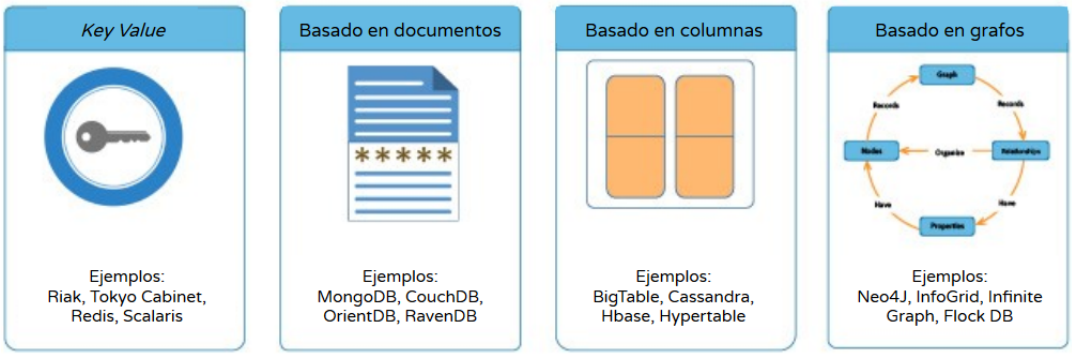


FIGURA 5.14: Tipos de bases de datos no relacionales
Fuente: Simplilearn

incluyendo muchos que tradicionalmente funcionarían sobre bases de datos relacionales.

- **NoSQL orientada a columnas:** Funcionan de forma similar a las bases de datos relacionales, pero almacenando columnas de datos en lugar de registros.
- **NoSQL de clave-valor:** Estas son las más simples en términos de estructuración de datos. Simplemente guardan tuplas que contienen una clave y su valor. Cuándo se quiere recuperar un dato, se busca dicha clave y se obtiene el valor.
- **NoSQL de grafos:** Basadas en la teoría de grafos, utilizan nodos y aristas para representar los datos almacenados. Son muy útiles para guardar información en modelos con muchas relaciones, como redes y conexiones sociales.

En la tabla mostrada a continuación se indican de forma resumida las características más importantes de cada uno de los modelos de base de datos no relacionales en términos de rendimiento, escalabilidad, flexibilidad y complejidad de manejo:

Tipo	Rendimiento	Escalabilidad	Flexibilidad	Complejidad
Clave-Valor	Alto	Alta	Alta	Muy baja
Columnas	Alto	Alta	Media	Baja
Documento	Alto	Alta	Alta	Baja
Columnas	Media	Media	Alta	Alta

CUADRO 5.1: Características de cada modelo de base de datos no relacional. **Fuente: UpWork Global Inc.**

Estas bases de datos, en su mayoría, cumplen con las propiedades ***"BASE"***. Este termino se considera opuesto al concepto ***" ACID"*** visto dentro de los modelos relacionales, puesto que es una filosofía de diseño de sistemas de datos que premia la disponibilidad sobre la consistencia de las operaciones [19]. Concretamente, este acrónimo esta compuesto por los siguientes conceptos:

- ***Basic Availability*:** Las bases de datos se centran en alcancar un nivel óptimo de disponibilidad, aún existiendo fallos dentro del sistema.

- ***Soft-State*:** Esta propiedad indica que el estado del sistema puede cambiar a lo largo del tiempo, incluso sin la existencia de operaciones de entrada. La consistencia de los datos debe ser gestionada por el propio desarrollador.
- ***Eventual Consistency*:** Este punto exige que en algún momento, los datos almacenados deben converger hasta un estado de consistencia. Sin embargo, no se asegura en que momento ocurrirá dicho estado.

El concepto ***"BASE"*** viene derivado del Teorema CAP. Esta idea sostiene que es imposible ofrecer simultáneamente las tres garantías (Consistencia, Disponibilidad y Tolerancia a la partición) [20]. Por tanto, hay que elegir dos de ellas:

- **Consistencia:** Todos los nodos tienen la misma información al mismo tiempo.
- **Disponibilidad:** Todos los clientes puedan leer y escribir, aunque se haya caído uno de los nodos.
- **Tolerancia a la partición:** El sistema continua funcionando a pesar de errores.

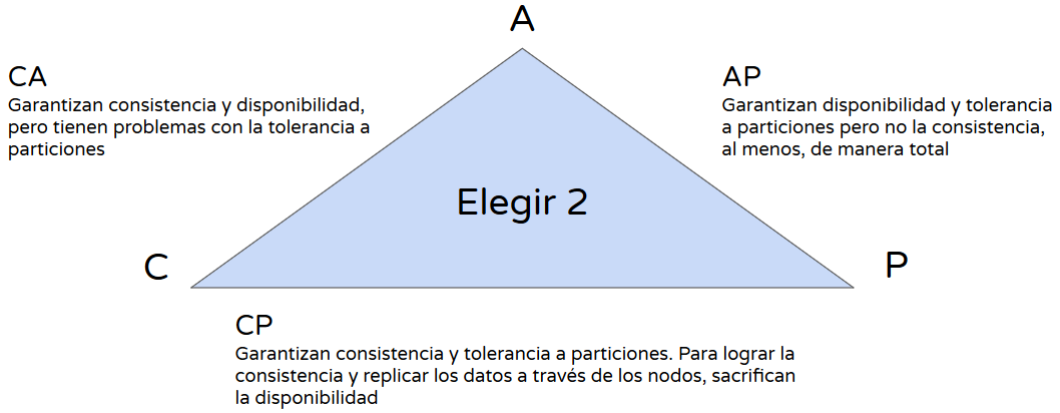


FIGURA 5.15: Modos de funcionamiento de las bases de datos NoSQL según el teorema CAP

Las definiciones mostradas en la ilustración anterior no son absolutas, puesto que la mayoría de servicios NoSQL pueden ser configurados para cumplir los diferentes modos de funcionamiento. Por ejemplo, la base de datos MongoDB funciona en modo CP por defecto, pero puede ser configurada para alcanzar mayores niveles de disponibilidad. El hecho de proveer esquemas flexibles permiten obtener un rendimiento óptimo en cada situación [21]. Un aspecto importante a tener en cuenta es que el rendimiento se mantiene constante independientemente del volumen de datos a manejar, tal y como se puede ver a continuación:

Este comportamiento contrasta con lo visto en la Figura 5.12, puesto que en ese caso el volumen de datos a manejar suponía un condicionante muy importante a la hora de determinar la eficiencia del sistema. Este es una de los motivos que han hecho que el uso de este tipo de base de datos esté tan presente dentro del ámbito profesional:

FIGURA 5.16: Rendimiento de una base de datos no relacional en función del volumen de datos. **Fuente: Datajobs**

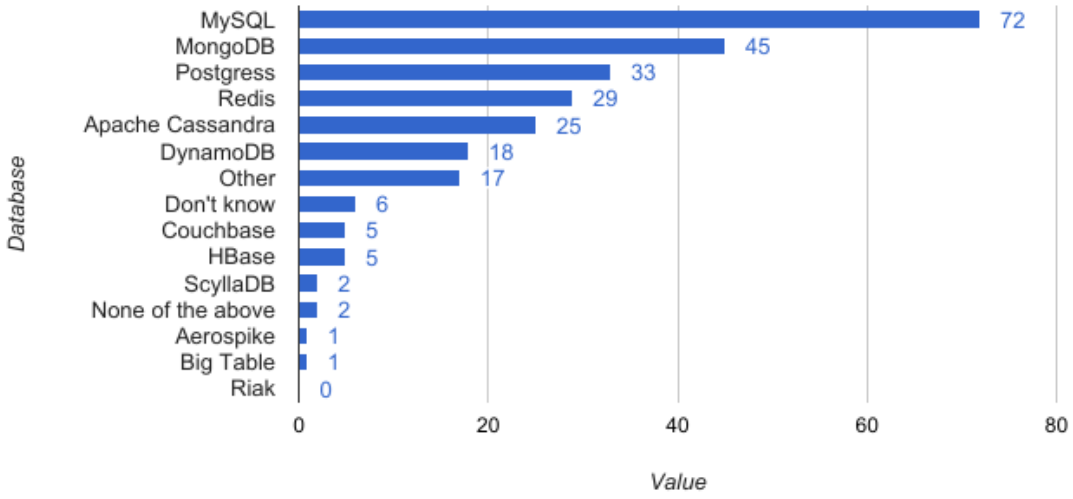


FIGURA 5.17: Porcentaje de uso de las diferentes bases de datos en entornos de producción. **Fuente: Amazon Web Services**

En lo referente a la escalabilidad en este tipo de modelos, se opta por la escalabilidad horizontal. Esto permite aumentar el rendimiento del sistema simplemente añadiendo más nodos e indicando al sistema cuáles son esos nuevos equipos disponibles. Además de esto, muchos sistemas NoSQL permiten utilizar técnicas como las que se presentan a continuación [22]:

- **Consultas *Map-Reduce*:** Consultas ejecutadas en todos los nodos simultáneamente (cada uno operando sobre una porción de los datos) y que finalmente reúnen los resultados antes de devolverlos al cliente.
- ***Sharding*:** Método para distribuir grandes cantidades de datos sobre diferentes equipos, de forma que no sea necesario replicar todos los datos en cada uno de los nodos y que la transferencia de información entre dichos nodos sea lo menos intensiva posible.

Si se relacionan todas las características descritas para los modelos de base de datos no relacionales con los criterios de selección definidos para el análisis, se puede

concluir lo siguiente:

- **Velocidad de procesado y rendimiento:** Una de las grandes ventajas de utilizar este tipo de modelos es poder obtener un rendimiento óptimo independientemente del volumen de tráfico a gestionar (véase Figura 5.16).
- **Escalabilidad:** La escalabilidad aportada por los modelos no relacionales es buena en términos generales. El hecho de apostar por una escalabilidad horizontal va a permitir obtener un rendimiento elevado en el sistema con la inclusión de nuevos equipos formando un *cluster*. Sin embargo, hay que considerar que el hecho de adquirir nuevos equipos para aumentar la capacidad del sistema puede suponer una inversión elevada, además de que la configuración para que todos los nodos funcionen de manera conjunta resulta compleja[4].
- **Consistencia de los datos y fiabilidad:** Los modelos no relacionales se caracterizan por dar un servicio más centrado a la disponibilidad que la consistencia de las operaciones. Sin embargo, tal y como se ha visto en el Teorema CAP (véase Figura 5.15), las bases de datos pueden ser configuradas para proveer unas características u otras. Al no hacer uso de transacciones *ACID*, el nivel de fiabilidad ofrecido no es equiparable al aportado por modelos relacionales, pero es suficiente para ofrecer las garantías requeridas en el servicio.

Comparativa entre SQL y NoSQL

En este apartado se va a proceder a realizar un breve resumen que describa las características fundamentales, así como ventajas y desventajas de cada uno de los tipos de bases de datos, de forma que se pueda facilitar la selección de la alternativa más favorable para el desarrollo del proyecto. La tabla que se presenta a continuación refleja los aspectos más relevantes que caracterizan tanto a las bases de datos relacionales como a las no relacionales.

Características	Bases de datos NoSQL	Bases de datos SQL
Rendimiento	Alto	Medio
Confiabilidad	Media	Alta
Disponibilidad	Alta	Alta
Consistencia	Media-Baja	Alta
Almacenamiento de datos[5]	Muchos datos	Pocos datos
Escalabilidad	Alta	Media

CUADRO 5.2: Comparativa entre bases de datos SQL y NoSQL.
Fuente: PandoraFMS Monitoring Blog

[4]La existencia de varios equipos implica que debe haber una sincronización adecuada entre ellos. Esta es una de las razones que dificulta la correcta consistencia de los datos.

Observando estas características, y conociendo las tendencias de mercado, se puede concluir que las bases de datos NoSQL suponen un gran avance para el tratamiento de grandes cantidades de información, y cada vez están llegando a unos niveles de confiabilidad más altos. Por otro lado, las bases SQL siguen muy presentes dentro de los sistemas de *backend*, con una fiabilidad alta y un buen rendimiento en la mayoría de los escenarios.

5.2.2. Almacenamiento seguro: Técnicas de securización

Los datos almacenados y gestionados por las empresas del sector del *e-commerce* suponen un activo muy importante para ellas. Entre esos datos se encuentra principalmente información referente a clientes (nombre, número de teléfono, direcciones, métodos de pago...etc.) así como contenido que tiene valor como propiedad intelectual (en este caso, licencias de *software*).

En este apartado se van a presentar una serie de alternativas que se pueden considerar a la hora de securizar los datos almacenados dentro de los sistemas de información. Los mecanismos mostrados pueden ser complementados con las técnicas de protección en el acceso vistas en el apartado 5.1 para ofrecer un nivel de seguridad más rotundo. A la hora de seleccionar aquella alternativa que resulte más adecuada para el modelo, se van a tener en cuenta los siguientes criterios de selección [23]:

- **Nivel de protección ofrecido:** Al igual que ocurría en la sección referente a la securización del acceso, es necesario valorar cual va a ser el nivel de seguridad que proporciona cada una de las alternativas. En este caso, debido a que es posible que se hayan incluido mecanismos de seguridad previos, hay que considerar el beneficio que aporta el añadir un nivel más de protección al securizar el propio almacenamiento de los datos.
- **Facilidad de implementación:** Otro punto a tener en cuenta es el nivel de dificultad que supone implementar un mecanismo de seguridad a nivel de almacenamiento de datos. En la mayoría de ocasiones, un método basado en una solución *software* es suficiente para poder aportar la seguridad deseada. Sin embargo, puede darse el caso de que se requiera de equipamiento adicional para soportar las funciones de seguridad, lo que hace que la instalación sea más compleja y también suponga un sobrecoste en la inversión necesaria.
- **Efectos sobre el rendimiento:** La inserción de mecanismos de seguridad, al igual que ocurría en la parte relacionada con el acceso, puede suponer una reducción en el rendimiento ofrecido por el sistema. Por tanto, hay que considerar la elección de una solución equilibrada que tenga en cuenta los posibles mecanismos de seguridad que se han utilizado con anterioridad para proteger el acceso a los sistemas de información.

[5] Considerando optimización en función de la cantidad de datos

FIGURA 5.18: Criterios de selección sometidos a estudio

Sin implementación de cifrado

Esta decisión resulta poco conveniente puesto que la protección de los datos viene dada unicamente por las técnicas de control de acceso que hayamos implementado. En caso de producirse una brecha de seguridad y que un usuario no autorizado pueda acceder al sistema, este va a ser capaz de leer y copiar toda la información almacenada dentro de la base de datos. El sistema de roles puede prevenir que el atacante realice determinadas operaciones sobre la base de datos, pero por norma general, esto no es suficiente para que la información no se vea comprometida.

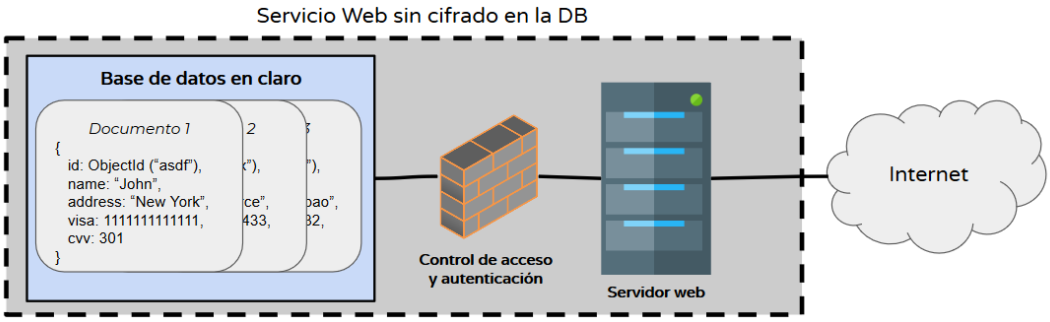


FIGURA 5.19: Servicio web sin securización en el almacenamiento

En relación con los criterios definidos con anterioridad se puede concluir las siguientes características de esta alternativa:

- **Nivel de protección ofrecido:** El nivel de seguridad de esta alternativa depende en exclusiva de los mecanismos de control de acceso y autenticación que se hayan prefijado en los puntos previos. Esto hace que esta solución sea muy pobre en términos de seguridad, puesto que si un atacante es capaz de sobrepasar esos controles, puede tener acceso a toda la información contenida dentro de la base de datos.
- **Facilidad de implementación:** Esta solución no conlleva ninguna complejidad más allá de los mecanismos de seguridad instalados en el control de acceso y autenticación.

- **Efectos sobre el rendimiento:** De la misma forma, el hecho de no implementar ningún mecanismo de seguridad a nivel de almacenamiento de datos provoca que el rendimiento pueda ser unicamente limitado por los sistemas de seguridad implementados en otros puntos.

Cifrado de la información

Esta alternativa propone el uso de técnicas criptográficas para poder ocultar los datos y que no sean legibles por usuarios externos. De esta forma, en caso de producirse un ataque que vulnere los sistema de control de acceso instaurados en el sistema, la confidencialidad de la información no será quebrantada, al menos de forma inmediata. La estrategia para la encriptación de los datos puede ser diseñada de distinta forma en función de diversos factores [24]. Algunos de estos son:

□ Nivel de encriptación

Se pueden considerar diferentes niveles en los que establecer la encriptación de los datos almacenados. Como nivel entendemos tanto el punto del sistema en el que se realiza el procesamiento para encriptar los datos, como la selección realizada a la hora de encriptar la información (todos los datos, solo información sensible...etc.):

- **Encriptación a nivel de disco:** El cifrado de los datos a este nivel pretende securizar la información al nivel más bajo posible (nivel *hardware*) y evitar el robo de información debido a la sustracción de los sistemas de almacenamiento (discos duros, memorias flash...etc.). Desde la perspectiva de la base de datos, supone una ventaja puesto que el proceso es totalmente transparente para el sistema, lo que permite no tener que hacer modificaciones a niveles más altos para ofrecer seguridad.

  Sin embargo, debido a que el soporte de almacenamiento no tiene conocimiento de la base de datos o sus estructuras, no es capaz de definir diferentes estrategias de seguridad dependiendo de los usuarios/roles. La encriptación selectiva de los datos en este sistema puede llegar a ser peligrosa puesto que, aunque cifremos la información correspondiente a la base de datos, existen ficheros de *logs* o cachés que pueden revelar información sensible.

- **Encriptación a nivel de base de datos:** Este sistema asegura que la información es securizada tanto cuando se inserta, como cuando es solicitada de la base de datos. En este caso, la estrategia de cifrado puede ser definida en función de los usuarios y sus roles/privilegios, lo que permite que se puedan diseñar diferentes cotas de seguridad para cada uno de ellos. En relación con este punto, también se da la posibilidad de establecer distintos niveles de granularidad (tablas, columnas, campos...etc), de forma que solo sean procesados aquellos puntos que sean sensibles.

  El proceso de cifrado y descifrado puede suponer una degradación en el rendimiento general de la base de datos, puesto que las operaciones requeridas para este proceso requieren un consumo considerable de CPU y los datos van a ser descifrados en tiempo de ejecución dentro del servidor donde se aloja la base de datos.

Dentro del sistema también deben estar almacenadas las claves de descifrado, y estas deben estar correctamente protegidas para que posibles intrusos no sean capaces de acceder a ellas y se hagan pasar por administrador del sistema.

- **Encriptación a nivel de aplicación:** Esta solución plantea mover el proceso de cifrado/descifrado a las aplicaciones que se encargan de la generación de los datos. De esta forma, la información es introducida encriptada dentro de la base de datos, puesto que la propia aplicación se encarga de todo el proceso. En una posterior operación de lectura, los datos serán entregados tal cual han sido almacenados, y la aplicación de nivel superior se encargará de descifrarlos.

  Esta aproximación tiene la ventaja principal de separar las claves de encriptación de los propios datos que se encuentran almacenados de forma cifrada en la base de datos. Las claves no tienen que moverse de la parte de la aplicación, lo que facilita la seguridad.

Por otro lado, una desventaja que puede generar este sistema es que las aplicaciones necesitan ser modificadas para adoptar la solución, además de que el rendimiento puede verse comprometido. En términos de granularidad y gestión de claves, este sistema resulta ser muy flexible puesto que estos factores dependen principalmente de la lógica de la aplicación que diseñe el desarrollador.

Mediante la siguiente ilustración se pretende definir de manera gráfica el funcionamiento de los tres métodos explicados en este apartado:

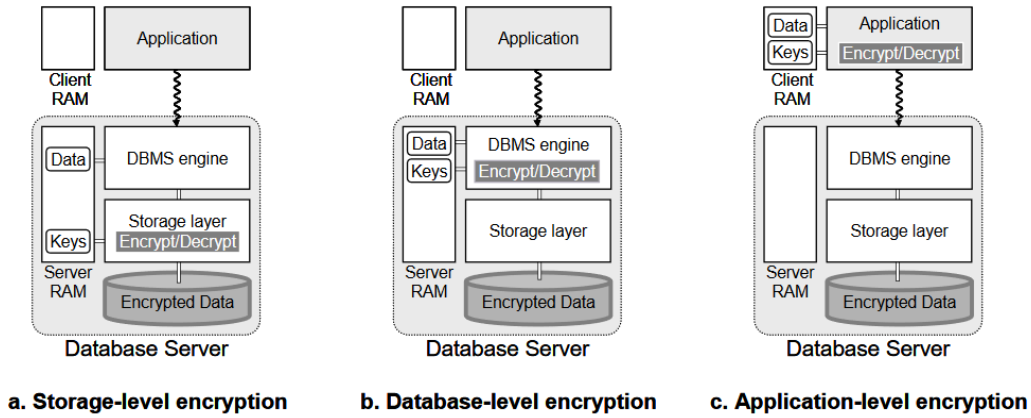


FIGURA 5.20: Niveles para encriptación de los datos.
Fuente: Encyclopedia of Cryptography and Security

□ Algoritmo utilizado y modo de operación

Independientemente de la estrategia utiliza, la seguridad de los datos encriptados depende principalmente del algoritmo utilizado, así como el tamaño de la clave y el modo de operación utilizado. Aunque se haga uso de un algoritmo de cifrado fuerte, como puede ser AES, si no elige un modo de operación correcto puede darse el caso de que parte de la información pueda verse comprometida [25].

Por ejemplo, si se utiliza un algoritmo en modo ECB, se van generar los mismos bloques de texto cifrado para entradas iguales. Eso implica que si hay datos repetidos dentro de nuestra base de datos, se pueden establecer patrones que los atacantes puedan utilizar para intentar descifrar la información. El contexto del sistema debe ser tenido en cuenta para evitar este tipo de vulnerabilidades. Hay que tener en cuenta que mucha de la información almacenada puede estar registrada dentro de la base de datos durante un largo periodo de tiempo[6]. En la figura que se muestra a continuación se incluye una representación de una imagen cifrada con el mismo algoritmo y utilizando diferentes modos de operación:

[6]**Ejemplo:** Las cuentas que usan los usuarios para registrarse, a menos que sean eliminadas por ellos mismos, se guardan de manera permanente en la base de datos.

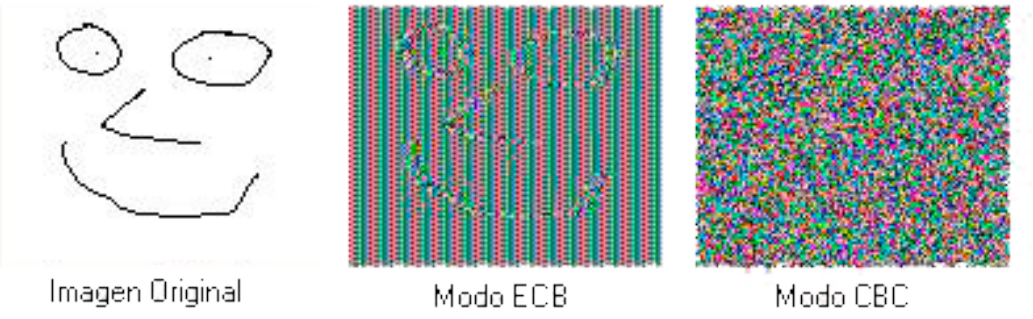


FIGURA 5.21: Comparación entre el modo ECB y CBC (Algoritmo Rijndael y llave de 128 bits). **Fuente: Microsoft**

Con el ejemplo que se muestra en la ilustración anterior se pretende demostrar la importancia en la selección de un modo de operación adecuado, puesto que este paso resulta tan importante como la implementación de un algoritmo fuerte y una clave de longitud elevada.

☐ Gestión de claves de cifrado/descifrado

Este punto referencia el modo por el cual se van a generar las claves criptográficas, y como estas van a ser gestionadas por el sistema. Debido a que los mecanismos de cifrado están basados en claves que se utilizan para cifrar y descifrar la información, la protección de la base de datos será buena siempre y cuando se haga una buena gestión de las claves.

Si se considera el caso de la encriptación a nivel de base de datos (consultar página 31), una solución que puede resultar sencilla en primera instancia es almacenar dichas claves de descifrado en una tabla/documento maestro que este fuertemente protegido por una *master-key* y que solo el administrador de la base de datos posea. El problema aparece cuando un usuario externo consigue suplantar la identidad de dicho administrador y, por lo tanto, es capaz de acceder a las claves y descifrar todos los datos sin ser detectado [24]. A modo de resolver estas problemáticas, se plantean los siguientes métodos de gestión de claves:

- **Uso de módulo de seguridad por *hardware* (HSM):** Las claves para el cifrado y descifrado van a estar almacenadas en el servidor y estas van a estar encriptadas por una clave maestra, la cual esta almacenada en dicho módulo *hardware*. Cuando hay que realizar operaciones sobre los datos, las claves son desencriptadas por el HSM utilizando su llave maestra, y estas son eliminadas de la memoria tan pronto como se acaben de realizar las operaciones de lectura y/o escritura dentro de la base de datos [26].
- **Uso de un servidor externo seguro:** Dentro de un equipamiento externo adicional se instala todo el *software* encargado de las tareas de seguridad. Este nuevo equipo debe gestionar los diferentes usuarios, roles, estrategias de encriptación y claves (pudiendo utilizar de forma conjunta un HSM). Un módulo adicional va a ser el encargado de realizar las comunicaciones correspondientes con la base de datos para autenticar a los usuarios y comprobar sus privilegios a la hora de realizar operaciones de lectura/escritura [27].

Se incluye a continuación una ilustración donde se muestra de manera esquemática el funcionamiento de cada uno de estos modelos de gestión de claves:

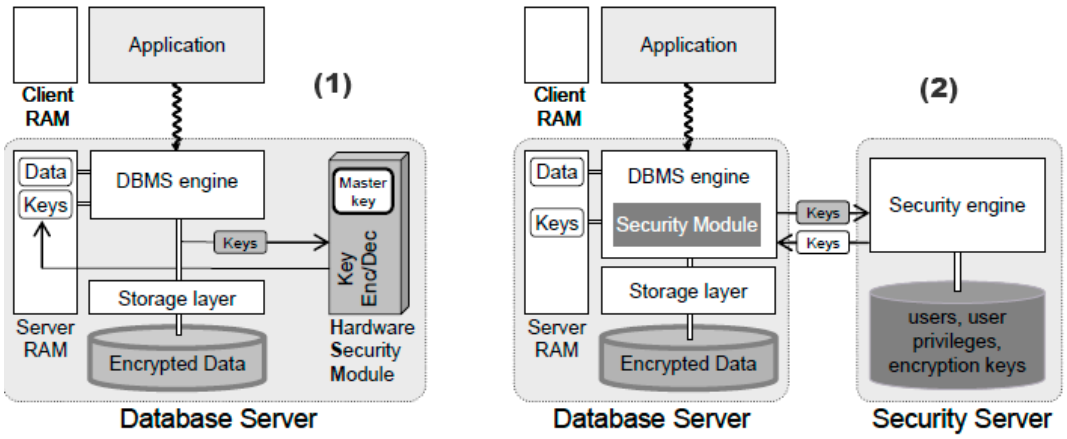


FIGURA 5.22: Gestión de claves: **(1)** HSM **(2)** Servidor Externo
Fuente: Encyclopedia of Cryptography and Security

☐ Implementación dentro de sistemas de bases de datos

Puesto que en apartados anteriores se han realizado comparativas entre bases de datos SQL y NoSQL, se va a continuar el estudio de ambas tecnologías en el campo de la seguridad, considerando que estrategias se pueden implementar en cada una de ellas. Mediante la siguiente tabla se han querido representar las características fundamentales de cada uno de los modelos de bases de datos:

		Base de datos relacional (MySQL)	Base de datos no relacional (MongoDB)
Nivel de encriptación	**Disco**	Sí, incluye TDE[7]	Sí, incluye el motor de almacenamiento seguro de *WiredTiger*
	Base de datos	Sí, uso de motor *InnoDB* para encriptar tablas, campos..etc.	No
	Aplicación	Soportado en la lógica de aplicación	Soportado en la lógica de aplicación
Algoritmo y Modo de operación	**Algoritmo**	AES, RSA, DSA	AES-256
	Modo de operación	CBC, Uso de *Salt* para la aleatorización	CBC, GCM, FIPS
Gestión de claves de cifrado	**HSM**	Sí	Sí
	Servidor externo	Si, (Protegity RSA, BSAFE, SafeNet)	Si, integración con sistemas externos con el protocolo KMIP

CUADRO 5.3: Seguridad en sistemas MySQL y MongoDB

[7]TDE es un modelo de encriptación muy similar a la encriptación a nivel de disco.

Tal y como se puede comprobar, independientemente de poseer modelos SQL o NoSQL, se pueden realizar implementaciones de cifrado potentes con la posibilidad de seleccionar diferentes estrategias de seguridad.

□ Resumen con características fundamentales

Una vez desarrolladas las diferentes estrategias que se pueden definir para el cifrado de los datos, así como las ventajas y debilidades de cada una de ellas, resulta interesante relacionar todos los conceptos vistos con los criterios de selección que se han definido al principio de la sección:

- **Nivel de protección ofrecido:** El nivel de seguridad depende en gran medida de la estrategia utilizada:
  - En lo referente al **nivel de encriptación**, todas las soluciones propuestas proveen una buena protección, siempre y cuando se tengan en cuenta los riesgos y consejos mencionados durante el desarrollo de las alternativas.
  - En lo referente al **algoritmo y modo de operación utilizado**, el uso de un algoritmo/clave fuerte debe ir siempre acompañado de un modo de operación adecuado que asegure una correcta protección de los datos (Figura 5.21).
  - La **gestión de claves** puede resultar un problema para la seguridad si no se considera una estrategia adecuada utilizando alguna de las herramientas explicadas con anterioridad (HSM o uso de servidor externo seguro).
- **Facilidad de implementación:** La solución propuesta puede llegar a ser muy compleja en términos de instalación y configuración. En caso de querer ofrecer un servicio con un nivel de seguridad elevado, hay que contemplar el uso de equipamiento externo para la gestión de claves, así como realizar las configuraciones que correspondan dentro del sistema de base de datos.

  En este caso, la selección de una estrategia basada en la encriptación en la lógica de aplicación puede facilitar en gran medida la implementación de un sistema de cifrado/descifrado, considerando además que esto también proporciona una seguridad adecuada al independizar el proceso de encriptación y el de almacenamiento.

  Tal y como se ha visto en la Tabla 5.3, los dos tipos de modelos de bases de datos contemplados para el almacenamiento de la información pueden ser configurados para establecer diferentes estrategias de seguridad, por lo que este punto no supone un inconveniente en ambas situaciones.
- **Efectos sobre el rendimiento:** La adoptación de mecanismos de cifrado y descifrado siempre tiene una repercusión sobre el rendimiento del sistema. De todas las soluciones planteadas, el uso de una encriptación a nivel de aplicación puede suponer una pequeña mejoría sobre el resto, puesto que cada aplicación realiza sobre su lógica la integración de los mecanismos de cifrado/descifrado. Las claves de cifrado/descifrado son separadas de los propios datos y no es necesario transferirlas desde la lógica de aplicación a la sección de almacenamiento.

5.3. Transferencia segura de la información

El último punto a analizar consiste en determinar el conjunto de mecanismos de seguridad que se pueden incluir dentro de la solución para poder ofrecer comunicaciones seguras entre los diferentes integrantes (Proveedor - Pequeño Distribuidor - Gestores de pago - Clientes). La transferencia segura de los datos supone un punto muy importante dentro de la arquitectura de la solución puesto que la mayoría de las transacciones e interacciones entre los elementos que conforman el sistema se van a realizar a través de una red no segura como es Internet. Por lo tanto, resulta imprescindible que la información a enviar esté correctamente protegida, con el objetivo de mantener la integridad, confidencialidad y disponibilidad de los datos:

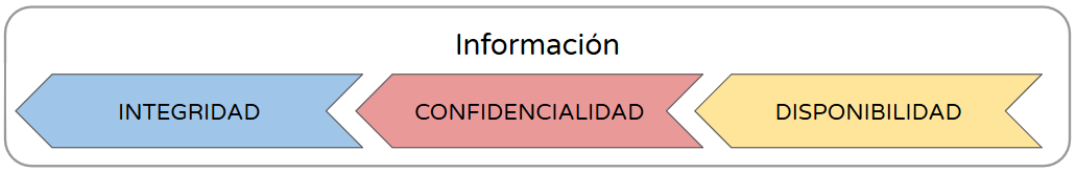


FIGURA 5.23: Información desde el punto de vista de la seguridad.

En lo referente a la transferencia de información a través de Internet, HTTPS supone un estándar para las comunicaciones seguras. La mayoría de proveedores de servicios actualmente existentes en la red facilitan el acceso a sus contenidos por medio de este protocolo. Por tanto, debido a la falta de alternativas, será una de las opciones a valorar dentro del modelo. Sin embargo, como soporte a esta tecnología se pueden utilizar otros mecanismos complementarios para proteger otros elementos en las comunicaciones (por ejemplo, la identificación de usuarios). Para estos mecanismos se van a definir los siguientes criterios de selección:

- **Nivel de protección ofrecido:** Hay que valorar el nivel de seguridad que ofrece cada alternativa frente a las otras opciones disponibles, además de conocer los beneficios que aporta su uso teniendo en cuenta que ya se está utilizando HTTPS dentro de las comunicaciones.
- **Facilidad de implementación:** Otro punto a considerar es el nivel de dificultad que supone que realizar la implementación de cada alternativa, tanto en términos de configuración, como en la propia lógica a añadir dentro de las aplicaciones web.
- **Efectos sobre el rendimiento:** Tal y como se ha mencionado en otros puntos, el hecho de incluir mecanismos de seguridad dentro del modelo tiene una implicación sobre el rendimiento general del sistema. En este caso hay que valorar cual de las alternativas propuestas presenta un comportamiento óptimo que minimice en el mayor grado posible su impacto sobre el rendimiento.

Los conceptos descritos a continuación están relacionados con el resto de ideas sobre seguridad que se han visto en apartados anteriores. Estas ideas aplicadas en conjunto con el resto de mecanismos de seguridad vistos anteriormente (securización en el acceso, almacenamiento seguro de los datos), permiten obtener un sistema seguro en todos los ámbitos.

5.3.1. HTTP Secure - HTTPS

HTTP Secure (HTTPS) es una extensión del protocolo HTTP utilizado tradicionalmente para la navegación web. Este protocolo proporciona mecanismos de seguridad para el cifrado y autenticación de mensajes mediante el uso de protocolos extras como TLS o SSL[8].

Tanto TLS como SSL son idóneos para su uso complementado con HTTP, puesto que son capaces de proveer seguridad en las comunicaciones aunque solo una de las partes esté verificada. Por norma general, las entidades que ofrecen servicios web disponen de certificados que permiten a los usuarios verificar la identidad de dichos equipos y evitar una posible suplantación de identidad [28] (*Man in the middle attack*).

El cifrado de los mensajes permite que todos los campos correspondientes al protocolo HTTPS puedan ser encriptados para asegurar la privacidad de los datos enviados. Esto incluye tanto la dirección URL solicitada, *query strings*, cabeceras y *cookies* (normalmente disponen de información que identifica a los usuarios). En la imagen mostrada a continuación se representa el procedimiento seguido al incluir los mecanismos de SSL/TLS en HTTP:

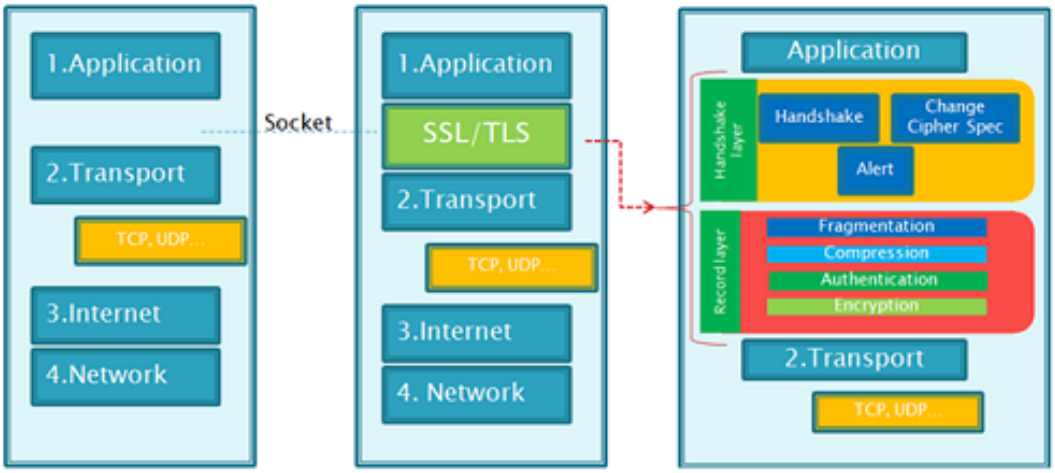


FIGURA 5.24: Procedimiento con SSL/TLS.
Fuente: DifferenceBetween.net

Si comparamos ambos mecanismos, es importante destacar que TLS es el sucesor directo de SSL, y, por lo tanto, parte de ser más seguro. Hablando más concretamente, SSL es vulnerable a varios tipos de ataques como *POODLE* o *BEAST*, mientras que con TLS esta carencias han sido solventadas [29]:

- ***POODLE Attack*:** Ataque que se aprovecha de intentos de conexión fallidos para forzar nuevas conexiones con un protocolo de comunicación más antiguo. De esa forma, un atacante podría ocasionar intencionadamente errores de conexión en protocolos seguros como TLS y forzar así el uso de SSL 3.0 para aprovechar vulnerabilidades existentes en esta revisión antigua del protocolo.

[8] Actualmente sustituido por TLS

- ***BEAST Attack*:** Este ataque vulnera la confidencialidad de las conexiones HTTPS al proveer un método para obtener en texto plano la información compartida en una sesión encriptada a través de SSL o TLS v1.0.
- ***DROWN Attack*:** Ataque que se aprovecha de aquellos servidores que disponen de TLS y soportan también SSL 2.0 para acceder a distintas vulnerabilidades presentes dentro de SSL y poder romper la seguridad del sistema.

Teniendo en cuenta el ámbito para la solución planteada, la mayoría de transacciones se realizarán por medio de peticiones web. Sin embargo, hay que tener en consideración que este tipo de protocolos no se utilizan unicamente para las comunicaciones web, sino que hay otros servicios donde también pueden ser implementados (véase apartado 5.1.4). La tabla que se muestra a continuación refleja de manera general las características de cada uno de los métodos planteados en este apartado:

	Diferencias	
	TLS	**SSL**
Año de lanzamiento	1999	SSL 2.0 en 1995 y SSL 3.0 en 1996
Bases	Basado en SSL 3.0 añadiendo mejoras	Solución inicial para proveer seguridad en web
Vulnerabilidades	BEAST para TLS 1.0	POODLE y BEAST entre otros
Nivel de seguridad	Más seguro	Menos seguro
Velocidad	Un poco más lento debido a su proceso de comunicación con algoritmos más complejos	Es más rápido al utilizar algoritmos más simples
Retrocompatibilidad	Compatibilidad hacia atrás con SSL	No compatible con TLS

CUADRO 5.4: Comparativa entre TLS y SSL

Uno de los puntos interesantes de utilizar HTTPS para las transacciones entre los elementos del modelo es que el resto de mecanismos pueden venir ligados a la protección ofrecida por esta solución. En este caso, si consideramos funciones como la autenticación para el control de acceso a determinadas funcionalidades, el hecho de enviar mensajes completamente cifrados nos permite incluir en la propia cabecera de las peticiones la información correspondiente al *login* de forma segura.

5.3.2. Gestión de sesiones por el cliente: *JSON Web Tokens* (JWT)

JSON Web Token (JWT) es un estándar abierto (RFC 7519) que define un mecanismo para transmitir información de manera segura entre los diferentes elementos de una comunicación, utilizando para ello objetos JSON. La información transmitida puede ser verificada debido a que es firmada digitalmente, ya sea utilizando un secreto (algoritmo HMAC) o con criptografía asimétrica (RSA). En cuanto a sus características, se cumple lo siguiente [30]:

- **Mecanismo Compacto:** Debido a su pequeño tamaño, los *tokens* pueden ser enviados a través de una URL, parámetros POST o dentro de una cabecera HTTP(s). Este nivel de compactación supone un buen rendimiento.
- **Mecanismo Autónomo:** El *payload* contiene toda la información del usuario, por lo que se evita tener que realizar consultas innecesarias a la base de datos para buscar esa información.

La idea planteada en este punto no supone una alternativa al uso de HTTPS, sino más bien un complemento que puede ser utilizado conjuntamente con el protocolo para añadir un grado mayor de seguridad a la hora de autentificar a los usuarios y comprobar que los datos transmitidos no han sido modificados por agentes externos.

Los *tokens* pueden ser tanto encriptados como firmados. En el primer caso, el objetivo que se pretende conseguir es ocultar la información enviada de elementos externos, mientras que con el firmado se pretende mantener la integridad de los datos[9]. Una ventaja de este sistema de transporte es que pueden aprovecharse los diferentes inicios de sesión realizados por las entidades del modelo para generar los *tokens* e identificar sus transacciones correspondientes. A partir de la información de sesión, el sistema genera un *token* formado por las siguientes partes[31]:

- **Header:** Se indica el tipo de autenticación utilizado (JWT), así como el algoritmo. Actualmente el uso de HS256 esta desaconsejado debido a que existen vulnerabilidades que permiten crear *tokens* validos de manera fraudulenta. Por tanto se recomienda el cifrado con RS256 el cual esta soportado y es seguro.
- **Payload:** La segunda parte contiene los datos que identifican al usuario, como puede ser su ID, nombre de usuario...etc.
- **Firma:** Se genera con las secciones anteriores, incluyendo un secreto o clave de cifrado dependiendo del algoritmo, y sirve para validar que el contenido no haya sido alterado.

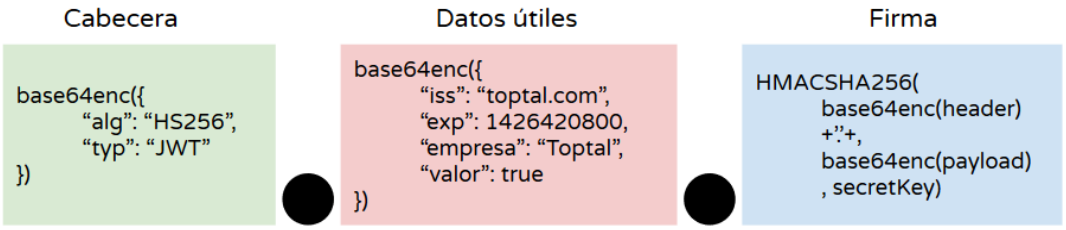


FIGURA 5.25: Estructura de JWT. **Fuente: TopTotal**

[9]La firma certifica que el integrante que dispone de la clave privada es el único que ha podido generar dicha firma

Tal y como puede verse en la imagen, la salida obtenida de este procedimiento son 3 *strings* codificados en Base64 y separados por puntos. Esto permite incluir la información dentro de las cabeceras HTTP(S) de manera más compacta que con otros mecanismos tradicionales basados en XML como SAML [30]. En la siguiente se muestra un ejemplo de *token*:

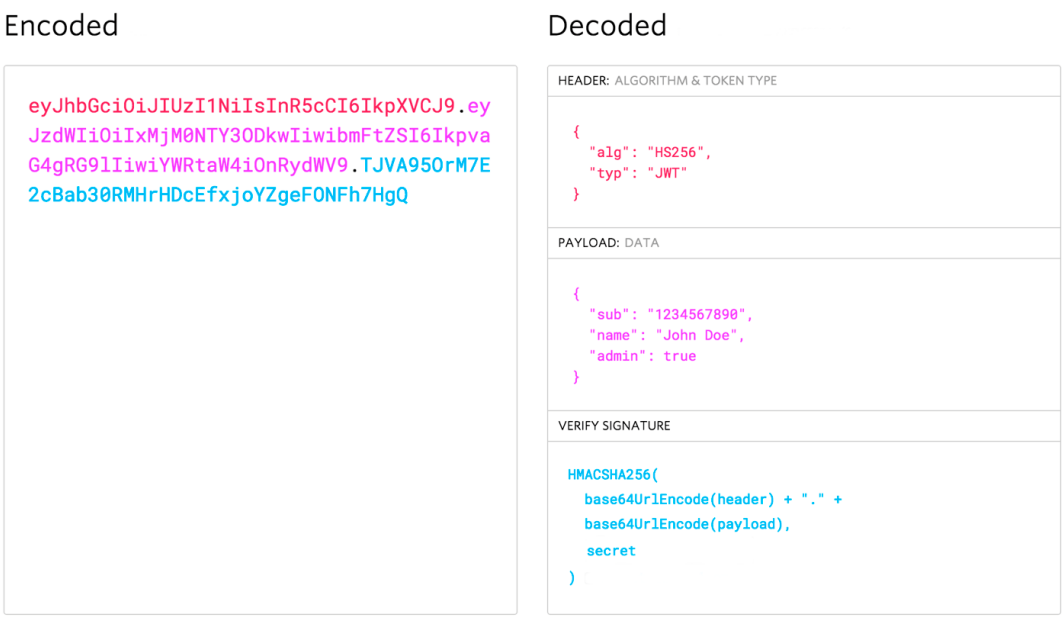


FIGURA 5.26: Ejemplo de *token* JWT. **Fuente: JWT.io**

En lo referente a la autenticación, cuando un usuario inicie sesión en un servicio introduciendo sus credenciales, se va a generar un *token* recogiendo esa información y será devuelto al cliente. Este *token* debe ser almacenado y se enviará en posteriores comunicaciones con el servidor. Al igual que ocurría con el almacenamiento de claves en las bases de datos (véase apartado 5.2.2), hay diferentes consideraciones de seguridad que deben tenerse en cuenta a la hora de realizar el almacenamiento de los *tokens*. A continuación se muestran las dos metodologías más utilizas a la hora de realizar la gestión del almacenamiento de los *tokens* [32].

- ***Web Storage*:** El *token* es almacenado de forma local por el navegador web. Esto permite que la implementación sea muy sencilla, pero la ejecución de código JavaScript malicioso puede propiciar que un atacante acceda a dicho repositorio y sustraer los *tokens* almacenados (*Cross-Site Scripting attacks (XSS)*).
- **Uso de *cookies*:** Mendiante el uso de de *cookies* se mantiene un control del estado de un usuario autenticado. La implementación puede resultar un poco más compleja que el caso anterior, sobretodo en caso de trabajar en comunicaciones entre dominios distintos, pero hay un mayor nivel de flexibilidad a la hora de controlar la seguridad (Forzado de HTTPS, Acceso restringido mediante JavaScript...etc.)[10].

[10]Mediante el uso de diferentes *flags* (HttpOnly, Secure...etc.) se pueden customizar los parámetros de seguridad referentes a las *cookies*

En el momento que el usuario quiera acceder a un sitio que requiera autenticación o que se encuentre restringido a determinados usuarios, esté deberá enviar por medio de las cabeceras HTTP(S) el *token*, concretamente, en el campo *"Authorization"* tal y como se ve en la siguiente imagen:

FIGURA 5.27: Cabecera HTTP con el *token* JWT. **Fuente: Auth0**

Este mecanismo de autenticación es completamente *stateless*, puesto que el estado del usuario no se almacena en ningún momento en la memoria del servidor. Las rutas que estén protegidas solicitarán que dentro de la cabecera esté el campo *"Authorization"* con la información referente al *token*. En caso de que sea correcto, se le devolverá una respuesta satisfactoria a su petición [30]. El procedimiento llevado a cabo por JWT puede ser resumido mediante la siguiente ilustración:

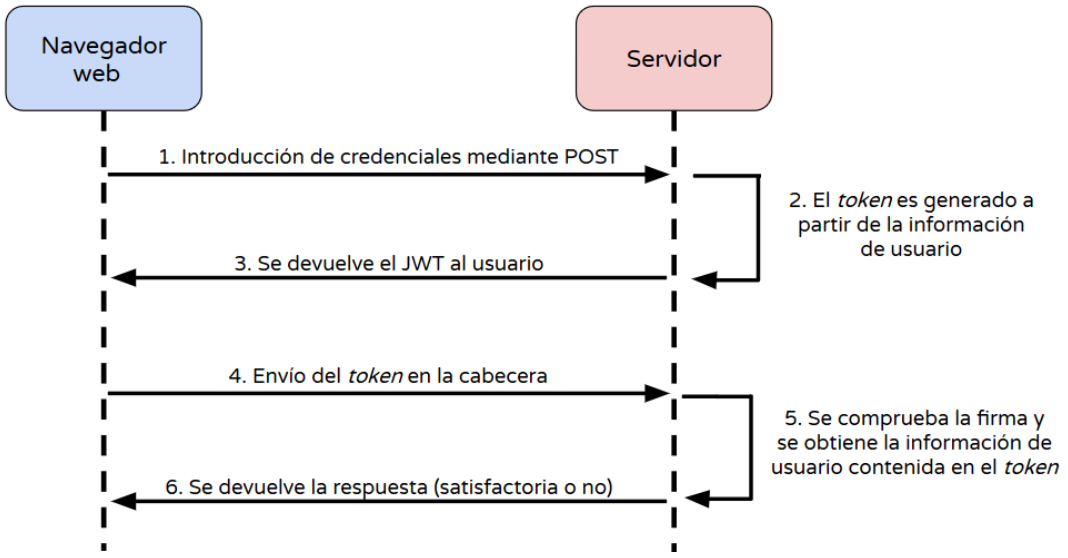


FIGURA 5.28: Funcionamiento de JWT. **Fuente: JWT.io**

Hay que tener en cuenta que la información referente al *token* es enviada en claro a menos que utilicemos técnicas adicionales para el cifrado de la información. Como se ha comentado anteriormente, esta solución se suele utilizar de manera conjunta con HTTPS para proveer un nivel de seguridad total en las comunicaciones.

Teniendo en cuenta las características que presenta esta tecnología, se van a relacionar estos conceptos con los criterios de selección que se han fijado anteriormente:

- **Nivel de protección ofrecido:** Esta opción proporciona un nivel de seguridad adecuado en términos de integridad de los datos. Sin embargo, si no se utilizan mecanismos complementarios como HTTPS para cifrar la información, cualquier agente externo va a ser capaz de leer los datos referentes a la identidad del usuario si captura el tráfico con un *sniffer*.

  Finalmente, tal y como ocurría con la gestión de claves en las bases de datos, es necesario una buena gestión a la hora de almacenar los *tokens* para que estos no sean leídos por agentes externos, puesto que en muchas ocasiones que los propios clientes hagan la gestión del almacenamiento de estos *tokens* puede suponer un problema de seguridad.

- **Facilidad de implementación:** La dificultad para implementar sesiones utilizando JWT resulta mayor que realizar la gestión de las mismas desde la parte del servidor. El uso de *cookies* resulta necesario para evitar posibles filtraciones de los *tokens*, y su implementación a nivel de *frontend* con una configuración de almacenamiento adecuada puede complicar la lógica de la aplicación.

- **Efectos sobre el rendimiento:** Una de las virtudes de esta tecnología es que la información referente a la sesión es muy liviana en comparación con la generada por otros métodos (SAML). Sin embargo, hay que tener en cuenta que hay existen otros factores que pueden afectar al rendimiento, como el algoritmo utilizado para generar la firma.

5.3.3. Gestión de sesiones en el servidor: El *middleware*

En referencia con la metodología desarrollada en el punto anterior, también existe la posibilidad de realizar la gestión de las sesiones de forma *stateful* dentro del propio servidor. De la misma forma que ocurría con los *JWT* almacenados en *cookies*, dentro de la información referente a sesión se pueden personalizar diferentes parámetros de seguridad (Forzado de HTTPS, acceso restringido mediante JavaScript, control de *proxys*, tiempo de vida...etc.), por lo que se da la posibilidad de definir diferentes estrategias de seguridad.

El modo de operación, en este caso, dista del comportamiento visto en el apartado anterior. Cuando al cliente se le envía una *cookie* para referenciar a la sesión, dentro de dicha *cookie* no se encuentra la información correspondiente a la sesión, sino que en su lugar es enviado un identificador (SessionID). A través de dicho identificador, el servidor es capaz de recuperar la información referente a la sesión y gestionar las peticiones oportunas según corresponda [33].

El almacenamiento es gestionado por el propio servidor, por lo que es necesario definir políticas para la protección de los datos referentes a las sesiones (véase apartado 5.2.2). En este caso, considerando un ejemplo basado en la solución *middleware* para NodeJS *"express-session"*, se dan las siguientes posibilidades[11] [34]:

- **Almacenamiento en memoria:** Es una solución que guarda toda la información de las sesiones en la memoria del propio servidor. Durante una etapa de desarrollo puede ser válido, pero en un entorno de producción resulta desaconsejable debido a problemas de seguridad (lectura de memoria) y de consistencia (dificultad para mantener sesiones en arquitecturas con varios nodos, no robusto frente a caídas y/o cambios...etc.).
- **Conexión con bases de datos MongoDB:** Almacenamiento de las sesiones en una base de datos MongoDB. En este caso, existe la posibilidad de utilizar diferentes conectores en el *middleware* que están diseñados con funciones especificas para la transferencia de datos de sesión a las bases de datos MongoDB, lo que facilita la implementación y resultan muy eficientes.
- **Conexión con bases de datos MySQL:** Almacenamiento de los datos en una base de datos MySQL. El sistema empleado es similar al utilizado con MongoDB, pero estas funciones están diseñadas para trabajar con sistemas de base de datos relacionales.

Una vez desarrolladas las ideas más importantes de esta técnica de gestión de sesiones, se procede a relacionar las características vistas con los criterios de selección que se han fijado para estas alternativas:

- **Nivel de protección ofrecido:** Esta opción provee la capacidad de definir estrategias de seguridad flexibles y que pueden llegar a ser tan seguras como las presentadas por medio de *JWT*. Por otro lado, que el propio servidor sea el responsable del almacenamiento y securización de los datos de sesión aporta un grado más de confianza, puesto que ya se ha demostrado que la gestión por parte de los clientes puede llegar a ser problemática en términos de seguridad y de configuración en el *frontend*.
- **Facilidad de implementación:** Este método, aún considerando la necesidad de gestionar un almacenamiento externo para las sesiones, no supone una complicación en términos de configuración en la parte de la lógica de aplicación. En lo referente a la securización de la base de datos, se pueden llevar a cabo las mismas prácticas utilizadas en los apartados anteriores, por lo que no se requiere diseñar estrategias de seguridad adicionales.
- **Efectos sobre el rendimiento:** El rendimiento va a venir condicionado por el desempeño que tenga el servidor web en sus comunicaciones con las bases de datos. En lo referente a la carga de trabajo, debido a que solo se envía un identificador y no datos de la sesión completa, también resulta ser una solución muy liviana.

[11] Existen desarrollos implementados para casi cualquier tipo de base de datos, ya sea relacional o no relacional. Sin embargo, los puntos descritos se han querido centrar en aquellos modelos más comunes y que se han estudiado a lo largo del documento.

5.4. Selección de las alternativas

Para finalizar con el apartado de alternativas, se incluye un resumen donde se especifica la ponderación realizada a la hora de evaluar cada una de las alternativas según los criterios propuestos. Con dicha evaluación, se seleccionarán aquellas alternativas que resulten más adecuadas para el desarrollo del modelo.

En primer lugar, por medio de la siguiente tabla, se van a valorar cada uno de los criterios definidos para cada elemento sometido a estudio. Esto va a permitir determinar que aspectos resultan más importantes a la hora de elegir una solución sobre otra:

	Criterios	Ponderación	Observaciones
Securización en el acceso a la base de datos	Equipamiento requerido	**25 %**	Una infraestructura compleja puede afectar de manera significativa al presupuesto y suponer un condicionante para que empresas pequeñas implanten el modelo
	Nivel de protección ofrecido	**50 %**	El sistema planteado debe ser lo suficientemente seguro para evitar posibles intrusiones o ataques
	Rendimiento proporcionado	**25 %**	La inserción de equipamiento para proteger el acceso puede generar cuellos de botella en la red interna
Almacenamiento estructurado: Tipos de bases de datos	Velocidad de procesado y rendimiento	**35 %**	El sistema utilizado debe ser capaz de gestionar grandes cantidades de datos de manera eficaz
	Escalabilidad	**20 %**	La infraestructura de red puede requerir ampliaciones en un futuro para dar un servicio óptimo a un mayor número de usuarios
	Consistencia de los datos y fiabilidad	**45 %**	Las operaciones realizadas deben ser consistentes para asegurar un correcto tratamiento de la información
Almacenamiento seguro: Técnicas de securización	Nivel de protección ofrecido	**40 %**	Gran parte de la seguridad del sistema se basa en elegir una buena estrategia para el almacenamiento seguro
	Facilidad de implementación	**30 %**	El método escogido debe ser correctamente configurado para evitar errores que causen brechas de seguridad en el sistema. Un sistema más complejo aumenta la probabilidad de encontrar más problemáticas.
	Efectos sobre el rendimiento	**30 %**	El uso de estas técnicas puede provocar un efecto muy negativo sobre el rendimiento del sistema
Transferencia segura de información	Nivel de protección ofrecido	**50 %**	La transferencia de datos sensibles a través de redes no seguras implica que los datos transferidos deben ser correctamente protegidos
	Facilidad de implementación	**20 %**	Un error de configuración puede suponer un problema de seguridad
	Efectos sobre el rendimiento	**30 %**	La información a transmitir debe ser liviana, aunque en muchos casos la capacidad en las comunicaciones va a venir dada por factores externos al proyecto.

CUADRO 5.5: Ponderación sobre los criterios de selección aplicados

Una vez se ha determinado una ponderación sobre los criterios de selección utilizados para analizar cada una de las alternativas, se procede a realizar una selección de las mismas en función a las características que se han descrito con anterioridad y las puntuaciones consideradas para cada criterio:

Securización en el acceso a la base de datos

	Equipamiento requerido	Nivel de protección ofrecido	Rendimiento proporcionado	Nota final
(1) *Hardware* dedicado para el *web-server* y la base de datos	9 x 25 %	3 x 50 %	5 x 25 %	**5**
(2) Uso de DMZ: Sistema básico	7 x 25 %	5 x 50 %	8 x 25 %	**6.25**
(3) Uso de DMZ: Solución basada en 2 *firewalls* y red segura	6 x 25 %	8 x 50 %	7 x 25 %	**7.25**
(4) Uso de DMZ: Solución basada en un *firewall* y varios interfaces de red	7 x 25 %	8 x 50 %	8 x 25 %	**7.75**

Como complemento a estas arquitecturas se puede considerar la aplicación de los siguientes mecanismos adicionales:

(5) Reemplazo de *hubs* por *switches*	8 x 25 %	7 x 50 %	8 x 25 %	**7.5**
(6) Uso de encriptación entre el servidor web y la base de datos: Integración nativa de SSL	6 x 25 %	4 x 50 %	5 x 25 %	**4.75**
(7) Uso de encriptación entre el servidor web y la base de datos: *SSH Port Forwarding*	5 x 25 %	4 x 50 %	5 x 25 %	**4.5**

CUADRO 5.6: Selección de alternativas para la securización en el acceso a la base de datos

Vistos los resultados, se puede considerar que la opción mas recomendada es la alternativa **(4) Uso de DMZ: Solución basada en un *firewall* y varios interfaces de red**. Esta solución es adecuada debido a que permite obtener buenos niveles de seguridad con el requisito de disponer unicamente de un *firewall*. Por otro lado, como complemento adicional puede resultar recomendable la utilización de *switches* sobre *hubs* **(5)**. En lo referente al uso de encriptación (alternativas **(6) y (7)**), resulta innecesario debido a que se van a utilizar técnicas de cifrado a otros niveles.

Almacenamiento estructurado: Tipos de bases de datos

	Velocidad de procesado y rendimiento	Escalabilidad	Consistencia de los datos y fiabilidad	Nota final
(1) Bases de datos relacionales (SQL)	7 x 35 %	5 x 20 %	8 x 45 %	**7.25**
(2) Bases de datos no relacionales (NoSQL)	9 x 35 %	7 x 20 %	6 x 45 %	**7.7**

CUADRO 5.7: Selección de alternativas para el almacenamiento estructurado

Tal y como se puede ver en la tabla mostrada anteriormente, el **uso de bases de datos no relacionales (2)** resulta más adecuado para nuestra solución debido a que su diseño está pensado para obtener un buen rendimiento y además poseen la flexibilidad suficiente para establecer diferentes estrategias de operación que permitan obtener un buen nivel de fiabilidad.

El tipo de base de datos no relacional a utilizar depende de la naturaleza de los datos que se van a almacenar. En este caso, como se trabaja con comercios electrónicos y se almacenan datos genéricos como usuarios y licencias, una base de datos orientada a documentos como **MongoDB** resulta ser una opción adecuada al poseer todas las características descritas anteriormente.

Almacenamiento seguro: Técnicas de securización

Debido a que se pueden definir diferentes estrategias de seguridad a la hora de proteger el almacenamiento de los datos, se van a analizar cada uno de los puntos que definen la estrategia de manera separada:

Nivel de encriptación

	Nivel de protección ofrecido	Facilidad de implementación	Efectos sobre el rendimiento	Nota final
(1) Sin implementación de cifrado	1 x 40 %	9 x 30 %	9 x 30 %	**5.8**
(2) Encriptación a nivel de disco	7 x 40 %	7 x 30 %	6 x 30 %	**6.7**
(3) Encriptación a nivel de base de datos	7 x 40 %	8 x 30 %	6 x 30 %	**7**
(4) Encriptación a nivel de aplicación	8 x 40 %	8 x 30 %	7 x 30 %	**7.7**

CUADRO 5.8: Selección de alternativas para el almacenamiento seguro: Nivel de encriptación

De todas las estrategias vistas en la tabla anterior, el uso de **encriptación a nivel de aplicación (4)** es la más idónea, puesto que el nivel de seguridad proporcionado es adecuado considerando la separación de funciones de cifrado/descifrado en la lógica de aplicación. Por otro lado, esto también ayuda a que el rendimiento sea más óptimo al limitar las bases de datos unicamente a funciones de almacenamiento.

Algoritmo y modo de operación

En este punto se pueden presentar diferentes alternativas de algoritmos y modos de operación, teniendo todos ellos un comportamiento bastante similar en cuestiones de rendimiento, protección e implementación (consultar Tabla 5.3 para ver modos y algoritmos compatibles con MongoDB).

En el caso concreto de este proyecto, debido a que se trabaja mayormente con información sensible, se van a utilizar **algoritmos de cifrado fuerte (AES)**, considerando un **tamaño de *key*** que sea lo suficientemente grande como para poder evitar ataques por fuerza bruta que rompan el cifrado **(128 bits)**. En lo referente al modo de operación ocurre algo similar. Una opción que resulta interesante para mantener un mayor grado de protección en los datos es utilizar **modos de operación que se basen en la aleatorización, como es el caso de CBC**. De esta forma, no se pueden obtener patrones para bloques de datos similares que puedan ayudar a los atacantes a romper el cifrado.

Gestión de claves

	Nivel de protección ofrecido	Facilidad de implementación	Efectos sobre el rendimiento	Nota final
(1) Uso de módulo de seguridad por *hardware* (HSM)	7 x 40 %	7 x 30 %	7 x 30 %	**7**
(2) Uso de un servidor externo seguro	8 x 40 %	5 x 30 %	6 x 30 %	**6.5**

CUADRO 5.9: Selección de alternativas para el almacenamiento seguro: Gestión de claves

De las dos alternativas consideradas, la opción **(1) Uso de módulo de seguridad por *hardware* (HSM)** resulta ser la más conveniente puesto que su implementación es más sencilla que en el caso de tu competidora, considerando además que el nivel de protección ofrecido sigue siendo adecuado para asegurar las claves.

Transferencia segura de información

	Nivel de protección ofrecido	Facilidad de implementación	Efectos sobre el rendimiento	Nota final
(1) HTTP Secure - HTTPS	8 x 50 %	8 x 20 %	9 x 30 %	**8.3**

Como complemento al protocolo se puede considerar la aplicación de los siguientes mecanismos adicionales de gestión de sesiones:

(2) Gestión de sesiones por el cliente: *JSON Web Tokens* (JWT)	7 x 50 %	6 x 20 %	8 x 30 %	**7.1**
(3) Gestión de sesiones en el servidor: El *middleware*	9 x 50 %	5 x 20 %	7 x 30 %	**7.6**

CUADRO 5.10: Selección de alternativas para el transferencia segura de información

La integración de un protocolo propietario para las comunicaciones entre los sistemas que componen el modelo puede llegar a ser muy complejo. Por lo tanto, se ha considerado que la única opción es utilizar **(1) HTTP Secure - HTTPS**, puesto que se trata de una solución madura utilizada de manera extendida para las comunicaciones en Internet y posee un rendimiento adecuado.

Por otro lado, el hecho de trabajar con información sensible correspondiente a los usuarios que utilizan el servicio, hace que se tengan que considerar metodos para la protección de las sesiones. En este caso, la alternativa más adecuada es la **(3) Gestión de sesiones en el servidor: El *middleware***, puesto que proporcionan un nivel de protección superior al realizar la propia gestión desde el servidor, y aunque la implementación puede ser un poco más compleja, el rendimiento ofrecido no dista mucho del proporcionado por la otra alternativa.

6 | Análisis de riesgos

El desarrollo de un proyecto de ingeniería siempre conlleva asumir una serie de riesgos. El proyecto presentado en este documento está centrado en el ámbito de las telecomunicaciones y la ciberseguridad, por lo que la mayoría de riesgos que se exponen a continuación están estrechamente vinculados con el servicio ofrecido a los usuarios, así como la protección de los datos a transmitir. Por lo tanto, antes de definir una solución concreta en base a las alternativas presentadas con anterioridad, resulta necesario estudiar los riesgos que pueden derivarse del desarrollo de este proyecto. A continuación, se resumen los diferentes puntos a tener en cuenta:

Riesgos externos

Los riesgos externos tienen su origen fuera el equipo del proyecto. Esto significa que hasta cierto punto no pueden ser controlados por la propia empresa. Destacan:

- **(1) Aparición de otra solución comercial:** Puede llegar a darse el caso de que una empresa dedicada al comercio electrónico que tenga gran influencia en el mercado diseñe una solución similar a la que se plantea en este proyecto. Si esto ocurre, muchas empresas del sector comenzarían a utilizar su solución, dejando el trabajo presentado en un segundo plano.
- **(2) Cambios en librerías o *APIs*:** El hecho de utilizar herramientas de terceros para incluir determinadas funcionalidades en el modelo implica que es posible que se produzcan cambios sobre las mismas para añadir características o resolver errores presentes. El modelo debe ser implementado considerando siempre las últimas revisiones de dichas librerías y *APIs* externas.
- **(3) Descubrimiento de vulnerabilidades:** Puede darse el caso de que se descubran nuevas vulnerabilidades de seguridad en los protocolos y tecnologías presentes dentro del modelo. Esto obligaría a todas las entidades que utilicen esta solución a realizar cambios dentro de la lógica de sus servicios web para hacer frente a esas amenazas y evitar así posibles ataques.

Riesgos internos

- **(4) Pérdida de datos:** La deterioración de los equipos, especialmente los que se encargan de gestionar y almacenar los datos referentes a licencias y usuarios, puede suponer que toda esa información sea perdida, con la consecuente repercusión económica y social.
- **(5) Brechas de seguridad:** Aunque este modelo pretende reforzar la seguridad para evitar posibles filtraciones de datos sensibles, puede darse el caso de que un fallo de seguridad provocado por un mal diseño, configuración o vulnerabilidad haga que los datos referentes a las personas y/o las licencias se vean comprometidos.
- **(6) Implementación poco eficiente:** El modelo presentado en el documento intenta ser lo más generalista posible para poder abarcar cualquier negocio electrónico que se dedique a la venta de licencias. Sin embargo, puede darse el caso de que empresas que decidan adoptar este modelo de comercialización utilicen herramientas y servicios que provoquen alguna incompatibilidad con el sistema propuesto. Esto puede causar desde pequeños problemas de rendimiento hasta la imposibilidad de adoptar el modelo a menos que se hagan los cambios necesarios en las arquitecturas actualmente presentes.

Análisis

Una vez determinadas las principales problemáticas del proyecto, se va a hacer uso de una herramienta de análisis de riesgos denominada ***"Matriz Probabilidad-Impacto"*** para establecer prioridades en cuento a los diferentes riesgos que pueden afectar al proyecto, considerando tanto la probabilidad de que ocurran, como el impacto que tiene sobre el proyecto que dichos riesgos den lugar.

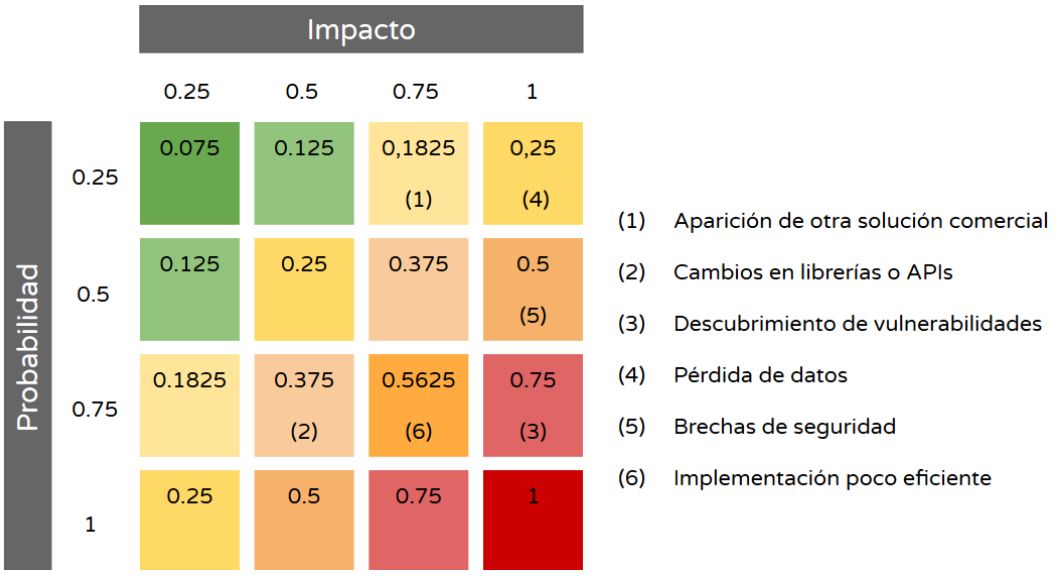


FIGURA 6.1: Matriz probabilidad-impacto del proyecto

Plan de prevención

El último paso a realizar en el proceso de análisis de riesgos consiste en definir un plan de prevención que permita minimizar la probabilidad de que los riesgos ocurran, así como establecer un protocolo de actuación en caso de que uno de los riesgos mencionados de lugar. Para cada uno de los riesgos anteriormente citados se establecen las siguientes acciones:

- **(1) Formación constante y establecimiento de acuerdos comerciales:** Una medida de previsión es estar constantemente informado sobre las actividades y proyectos realizados por empresas de referencia dentro del sector del comercio electrónico. En caso de que tengan la intención de lanzar una solución relacionada con este proyecto, debería intentarse establecer acuerdos comerciales con esas empresas para iniciar un desarrollo conjunto y obtener mayor reconocimiento de cara al mercado.
- **(2) Revisiones constantes del modelo:** La posible solución a este riesgo consiste en revisar periódicamente el modelo y todas las herramientas incorporadas en el mismo con el fin de detectar posibles fallos y actualizar aquellas librerías o *APIs* que lo requieran.
- **(3) Diseño modular de la arquitectura de red:** El hecho de detectar una vulnerabilidad dentro del sistema podría suponer un problema grave dentro de la solución propuesta. Por tanto, se pretende hacer un diseño lo más modular posible, diferenciando cada una de las partes implicadas en la comunicaciones, y utilizando técnicas de seguridad distintas en cada punto para evitar que un problema comprometa a toda la arquitectura propuesta.
- **(4) Uso de equipamiento de *backup*:** Uno de los puntos contemplados a la hora de ofrecer un servicio web es tener un alto nivel de disponibilidad. Por tanto, hay que asegurar que los servicios ofrecidos estén operativos el mayor tiempo posible y que la caída o rotura de un equipo no suponga la perdida de información. Para conseguir esto, lo que puede utilizarse dentro de la infraestructura de red de cada empresa es un sistema de *clusters* y sistemas redundantes que se encarguen de la replicación de la información almacenada (Bases de datos, *logs*, claves de cifrado/descifrado...etc.). En la imagen siguiente se muestra el proceso de replicación:

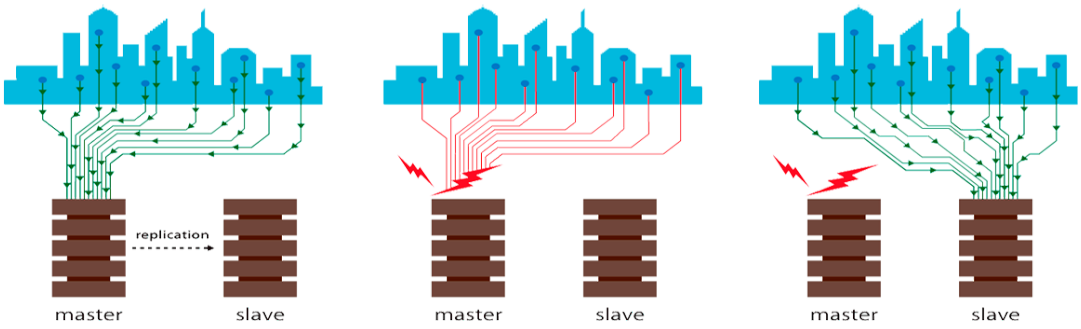


FIGURA 6.2: Replicación de contenidos para alta disponibilidad
Fuente: Vertabelo

- **(5) Prevención y respuesta ante filtraciones:** Esta idea se encuentra estrechamente relacionada con el punto **(3)** mencionado anteriormente. A modo de prevención hay que intentar desarrollar un sistema modular aplicando diferentes técnicas de seguridad y aportando siempre las últimas revisiones de los algoritmos y protocolos de encriptación. En caso de producirse un ataque a la seguridad del sistema, sería necesario contactar con los usuarios afectados y dar de baja los servicios y licencias que hayan sido comprometidas.

- **(6) Entrega de documentación referente al modelo diseñado:** Las empresas deben conocer el modelo y su modo de operación para saber de antemano que incompatibilidades pueden surgir durante la implementación del mismo. Para evitar problemas de este tipo, se recomendará siempre seguir las pautas y diseños especificados en el proyecto, sobretodo considerando la lógica de negocio, puesto que el equipamiento es más flexible en ese sentido.

7 Descripción de la solución

En el siguiente apartado se procede a especificar el diseño que se ha definido para la solución del proyecto. En la imagen siguiente se muestran las componentes principales de la solución diseñada, así como la relación existente entre ellas:

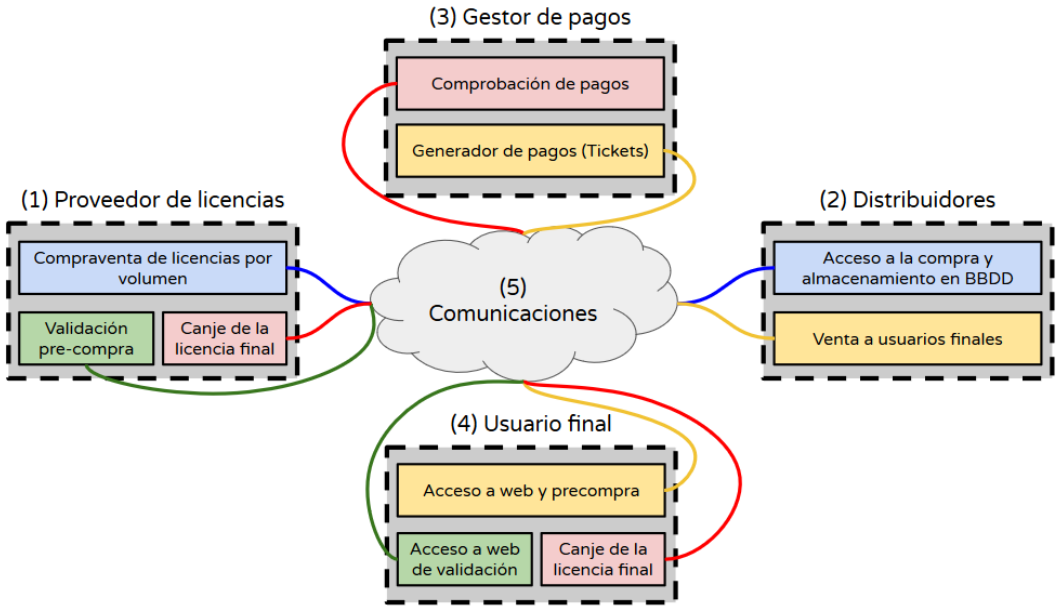


FIGURA 7.1: Escenario general de aplicación

Tal y como se puede apreciar en la imagen anterior, el modelo planteado incluye diferentes agentes con funcionalidades que permiten el desarrollo de un proceso de gestión y compra de licencias seguro. A continuación se describe de forma breve las funcionalidades de cada uno de ellos:

- **(1) Proveedor de licencias:** Empresa que se dedica a la creación de *software* y/o vender licencias en gran volumen a pequeños distribuidores. Esta parte actúa como un agente de confianza que los usuarios finales pueden utilizar para verificar el estado de las licencias previo pago.
- **(2) Distribuidor:** Comercio *online* que compra licencias en gran volumen al **(1) Proveedor de licencias** y posteriormente las vende a **(4) Usuarios finales** a un precio más económico.

- **(3) Gestor de pagos:** Agente externo que se ocupa de realizar las operaciones de pago. Se constituye por una *API* cerrada que solo puede ser accedida por comercios autorizados. Permite a los **(4) Usuarios finales** realizar el pago de las compras realizadas, así como verificar el pago en la parte del **(1) Proveedor de licencias** para registrar definitivamente la clave adquirida.
- **(4) Usuario final:** Cliente que compra al **(2) Distribuidor** la licencia, y usa al **(1) Proveedor de licencias** para verificar su estado previamente al pago y registrarla tras el mismo.
- **(5) Comunicaciones:** Este punto describe la forma de realizar las transacciones entre los diferentes agentes que conforman el modelo propuesto.

A continuación, se explican de manera extendida las funciones y elementos que van a poseer cada una de las partes, las cuales se han separado en diferentes subapartados. Todas las funcionalidades y elementos que se describen a continuación serán utilizados posteriormente para definir un entorno de pruebas donde se pueda verificar y validar el correcto diseño del modelo.

7.1. Proveedor de licencias

Esta parte del modelo se centra principalmente en las funcionalidades que debería disponer la empresa que se encarga de la creación del *software* a vender y/o que distribuye licencias a pequeños vendedores. Para comenzar con la descripción del modelo, se incluye un esquema donde se han definido los diferentes elementos que componen la arquitectura diseñada para esta parte:

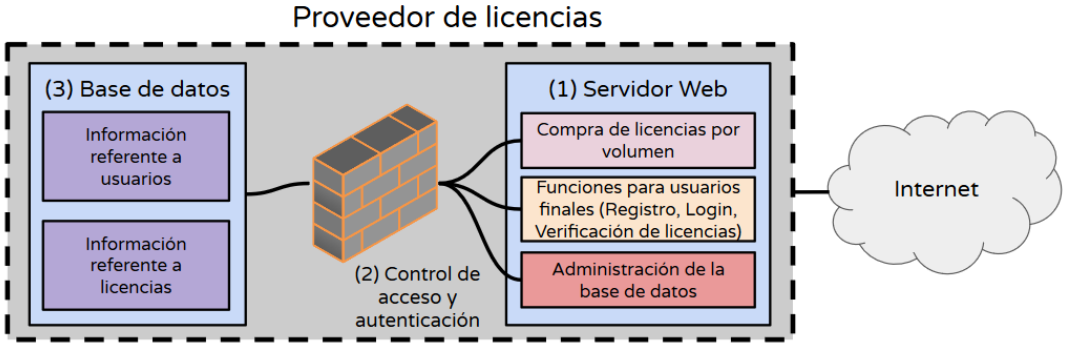


FIGURA 7.2: Propuesta de arquitectura para el proveedor de licencias

Tal y como puede apreciarse en la figura anterior, el sistema consta de tres partes fundamentales, el servidor web, la sección de control de acceso y autenticación, y la base de datos:

- **Servidor web:** Interfaz entre el proveedor y los clientes, aportando las funciones para la compra de licencias en caso de los distribuidores, así como la capacidad de verificar las licencias y canjearlas en el caso de usuarios finales.

- **Control de acceso y Autenticación:** Mecanismos de seguridad requeridos para evitar intrusiones en el sistema.
- **Base de datos:** La base de datos actúa como *backend*, donde se almacena toda la información relacionada con los usuarios y las licencias, y que alimenta a los servicios web mencionados anteriormente.

A continuación se describen de forma más detallada las características de cada una de las partes que conforman el proveedor.

7.1.1. Servidor web

El servidor web es el elemento que van a utilizar el resto de participantes de la red para poder realizar transacciones con el proveedor de licencias y utilizar sus servicios. Tal y como se venía adelantando en la Figura 7.2, dentro de la lógica de aplicación del servicio web se deben incluir diversas funcionalidades relacionadas con la gestión de usuarios y licencias. Estas se describen brevemente a continuación:

- **Funciones para usuarios finales:** Estas funcionalidades están centradas en proveer mecanismos para el registro, inicio de sesión y verificación de licencias para los usuarios.
- **Compra de licencias por volumen:** Lógica de aplicación centrada en proveer un sistema fiable y estructurado que permita a los distribuidores poder adquirir licencias en grandes volúmenes para su posterior venta a usuarios finales.
- **Administración de la base de datos:** Funciones que permiten a los administradores del sistema conocer el estado de las licencias disponibles así como añadir nuevas entradas.

Funciones para usuarios finales: Registro

Una de las primeras acciones a realizar para que los usuarios puedan utilizar el servicio es incluir dentro de la lógica de aplicación las funcionalidades relacionadas con el registro. Mediante la creación de una cuenta, los usuarios van a ser capaces de acceder al resto de funcionalidades dedicadas a la verificación y registro de licencias.

El mecanismo más sencillo e intuitivo para que los usuarios finales puedan registrarse en la plataforma es proveer un formulario donde introduzcan los datos correspondientes a su identidad. Concretamente, dentro del formulario se debe introducir tanto un nombre de usuario, como dirección de correo electrónico y contraseña. El nombre de usuario y el *email* deben ser únicos, es decir, no deben existir cuentas registradas anteriormente con un mismo nombre de usuario y/o correo electrónico. En lo referente a la gestión de contraseñas, resulta muy peligroso almacenarlas como un texto plano, por lo que dentro de la propia lógica de aplicación se realizará un procedimiento de *hash* antes de enviar la información a la base de datos.

El proceso de creación de cuentas no dista de los procedimientos seguidos en otras plataformas *online*, sean o no empresas dedicadas al comercio en línea. En la siguiente ilustración se muestra una propuesta de como debe ser el formulario de registro para los usuarios:

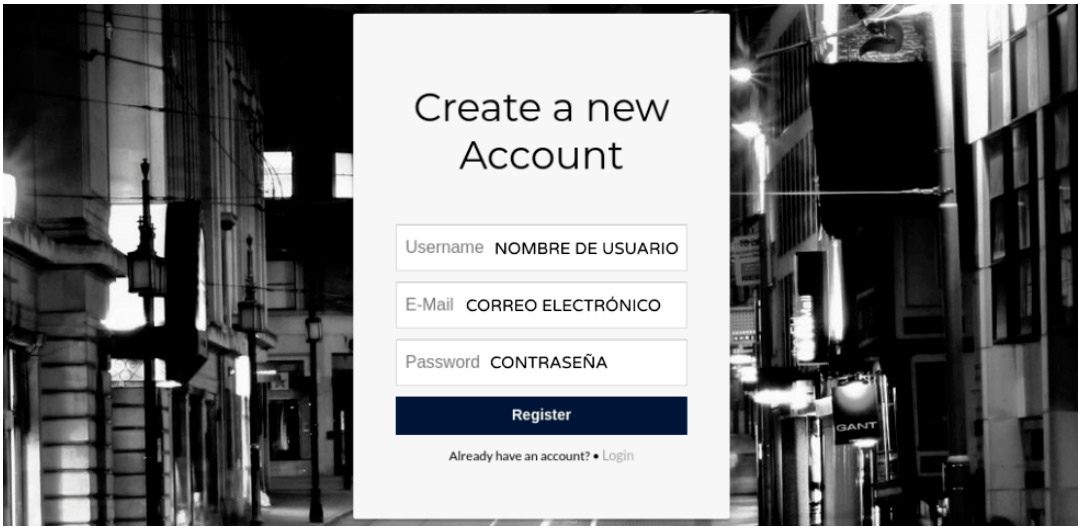


FIGURA 7.3: Formulario de registro

Funciones para usuarios finales: Inicio de sesión

El sistema tiene un comportamiento similar al descrito en el apartado anterior. Los usuarios requieren iniciar sesión para acceder al perfil y poder verificar y/o canjear las licencias adquiridas. El proceso de inicio de sesión se realiza por medio de un formulario muy similar al mostrado en el apartado anterior, tal y como se puede ver en la siguiente imagen:

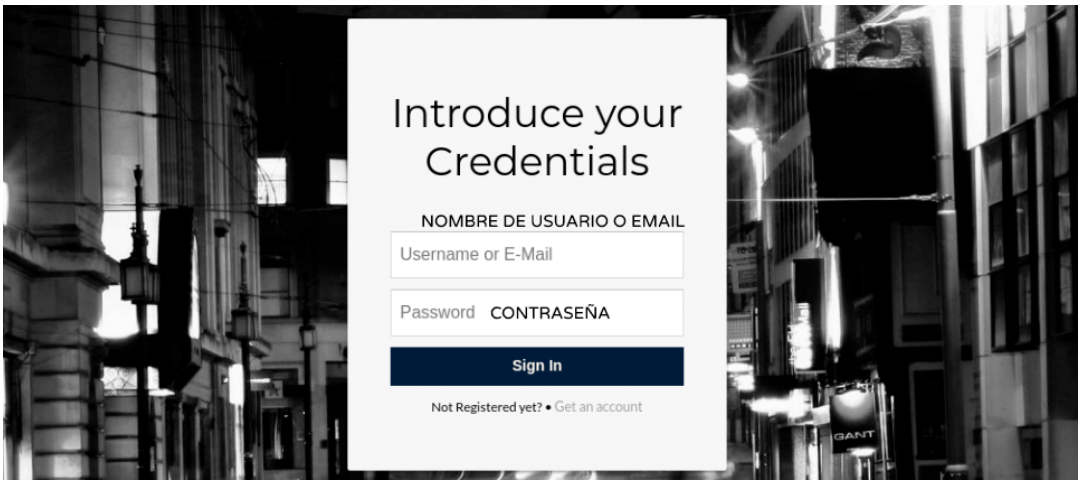


FIGURA 7.4: Formulario para el inicio de sesión

El usuario que quiera acceder a su perfil deberá introducir sus credenciales dentro del formulario mostrado anteriormente. El primer campo a rellenar permite la introducción del nombre de usuario o la dirección de correo electrónico utilizada para el registro. El segundo campo se corresponde con la contraseña. Al igual que ocurria en el apartado de registro, es necesario realizar un procedimiento de *hash* en la contraseña introducida para verificar la validez de la misma. La imagen que se presenta a continuación describe el procedimiento de manera esquemática:

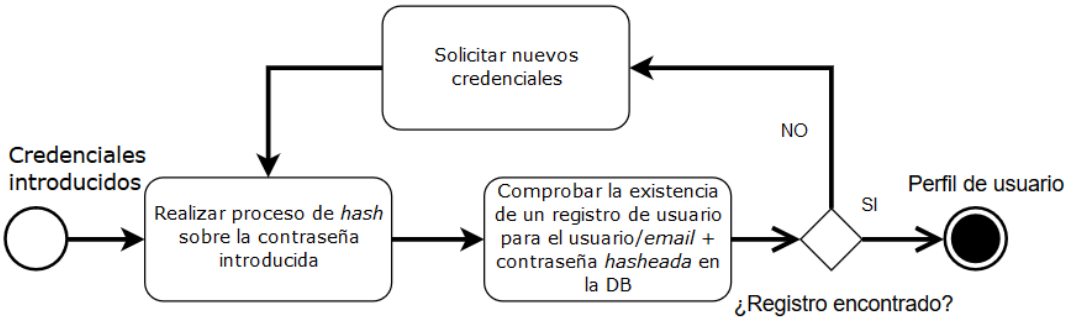


FIGURA 7.5: Diagrama de actividad para el inicio de sesión

Funciones para usuarios finales: Verificación y registro de licencias

Una vez el usuario ha introducido correctamente sus credenciales, este ya puede acceder a las funciones destinadas a la verificación y registro final de licencias. Para ello, dentro de la propia página de perfil se ofrecerán varias opciones, las cuales se presentan en la siguiente imagen:

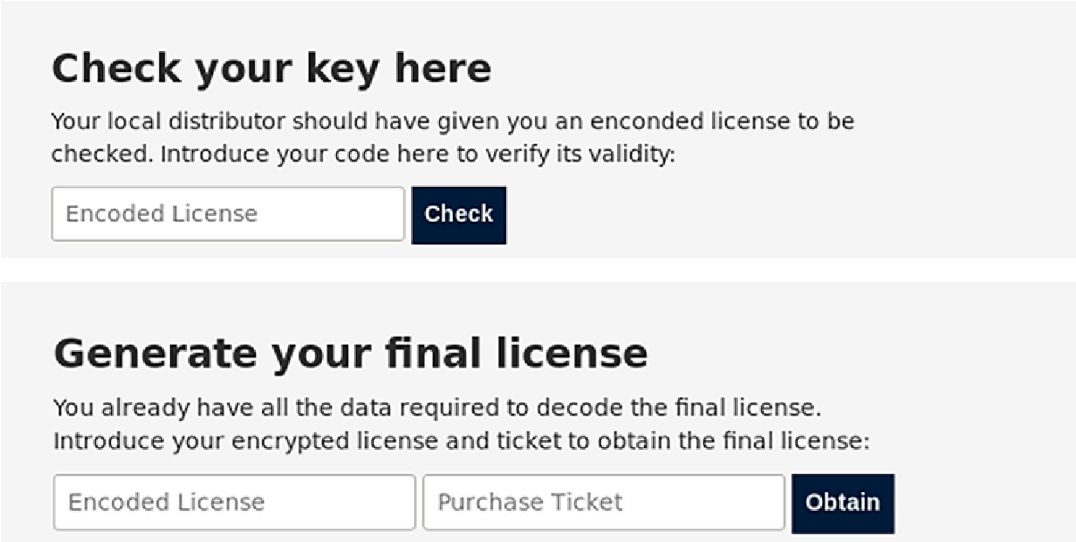


FIGURA 7.6: Funciones para la verificación y registro de licencias

A continuación se realizará la descripción detallada de cada una de estas funcionalidades:

□ Comprobación de la licencia

Esta función es muy relevante dentro del sistema, puesto que va a permitir al usuario conocer el estado de la licencia que va a adquirir, previamente a haber hecho el pago de la misma. A través del formulario mostrado en pantalla, el usuario introducirá la licencia encriptada que el distribuidor le ha otorgado antes del proceso de pago, para que realice su verificación. Tras realizar la consulta correspondiente, se entregará un mensaje de estado donde se pueden dar varias situaciones:

1 - No existe	2 - Registrada	3- Consultada	4 - OK
El usuario ha introducido una licencia que esta mal escrita o el distribuidor le ha entregado una licencia no válida	El distribuidor ha entregado una licencia al cliente que ha sido previamente vendida y que se encuentra registrada por otro usuario	La licencia que ha introducido el usuario la ha verificado otro con anterioridad[1].	La licencia no ha sido registrada ni consultada previamente.

CUADRO 7.1: Posibles estados en la verificación de una licencia previo pago

En caso de darse la situación 1 o 2, el usuario debería cancelar el pedido en el distribuidor para evitar un posible engaño. Por el contrario, si la verificación resulta satisfactoria, el usuario puede continuar con el proceso de pago con total seguridad. Este sistema, tal y como puede verse, permite al usuario tener un mayor nivel de garantías a la hora de realizar la compra de la licencia.

□ Registro de la licencia final

A través de esta funcionalidad el usuario final va a poder gestionar el registro de la licencia final. Para ello, tal y como puede verse en la Figura 7.6, el usuario debe introducir dos parámetros dentro del formulario:

- **Licencia encriptada:** Se corresponde con el código que se ha utilizado previamente para la verificación de la licencia y que entrega el distribuidor al cliente.
- **Ticket de compra:** Se trata de un código único generado a través de un método propietario por el gestor de pagos y que es entregado al usuario posteriormente a realizar el pago.

Tras introducir los parámetros, se va a realizar un procedimiento dentro de la lógica para verificar que los datos introducidos sean correctos, y en caso de ser así, generar la licencia final al usuario. El procedimiento seguido se puede ver en la siguiente diagrama:

[1]Esto puede ser debido a dos razones: (1) El cliente anterior canceló el pedido pero llegó a verificar la licencia. (2) La licencia ha sido vendida pero el usuario no ha canjeado la licencia final. Comprar la licencia en esta situación supone un cierto grado de incertidumbre.

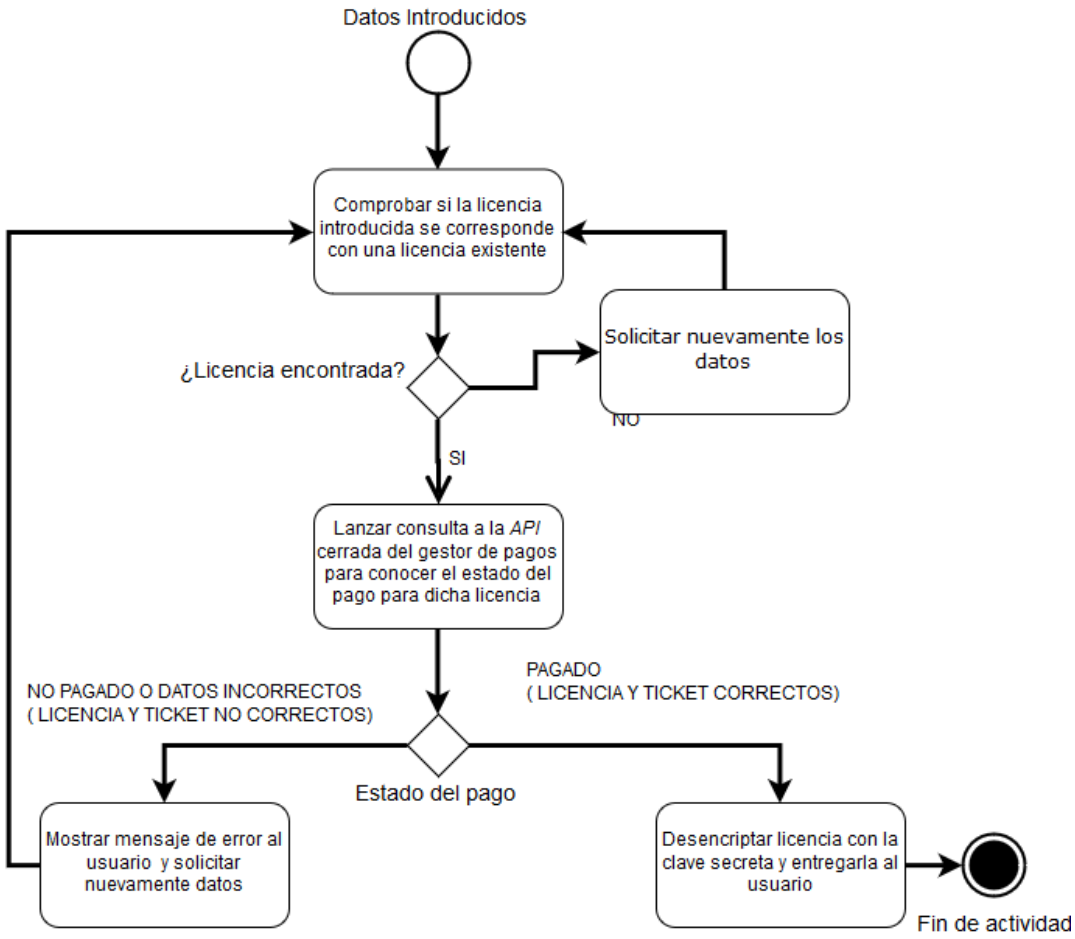


FIGURA 7.7: Diagrama de actividad para el registro de las licencias

Debido a que las funciones relacionadas con la verificación y el registro de la licencia requieren que el usuario este autenticado, se puede llevar un control sobre las mismas. En el momento que un usuario solicita la consulta o registro de una licencia, la información referente a la misma va a ser actualizada indicando la identidad del usuario y fecha para la operación realizada.

Compra de licencias por volumen

Esta parte incluye toda la lógica referente a la compra de licencias en grandes volúmenes por parte de los distribuidores para que posteriormente puedan venderlas a los usuarios finales a un precio más económico. El sistema está formado por una página con un formulario donde se deben indicar los diferentes datos referentes a la empresa que quiere realizar la compra de licencias. Entre ellos encontramos datos fiscales, método de pago...etc. Esta parte es más susceptible de modificaciones en función de los datos que quiera solicitar cada empresa a la hora de vender licencias en grandes volúmenes a otras organizaciones. Un ejemplo de formulario es el mostrado en la siguiente imagen:

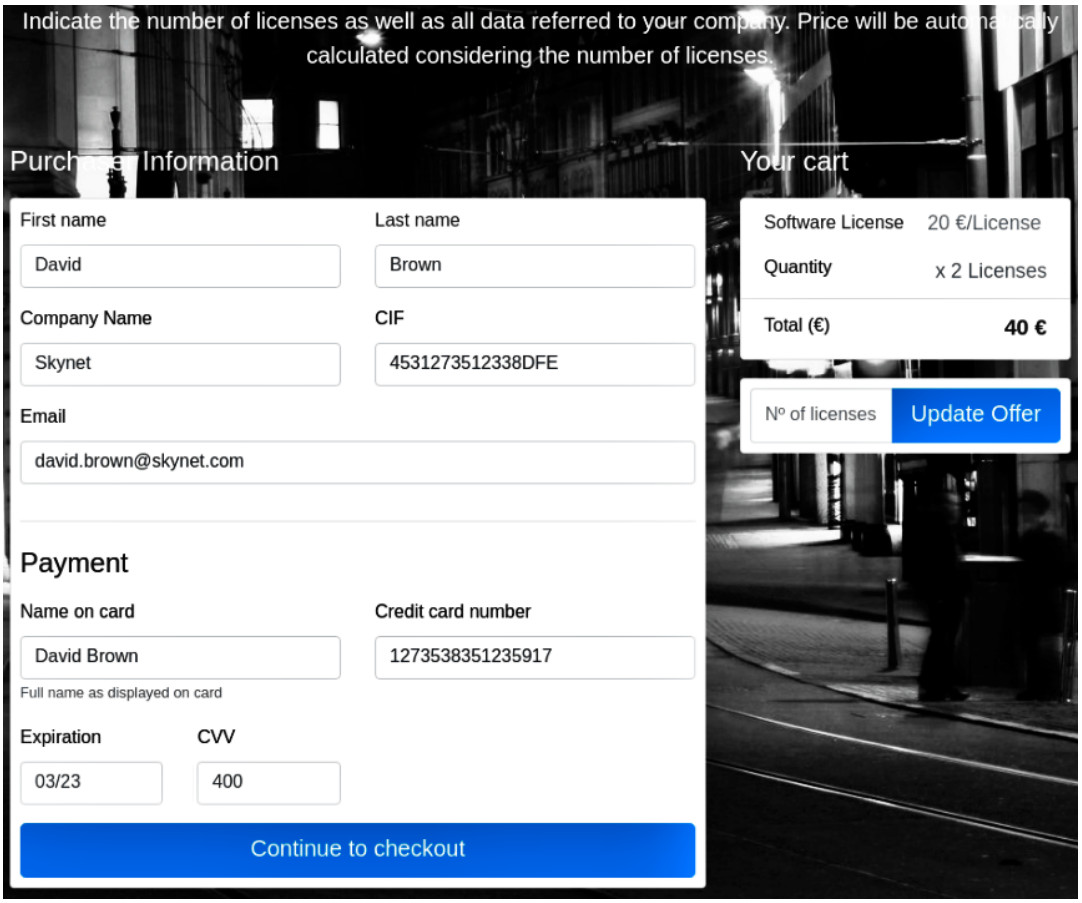


FIGURA 7.8: Ejemplo de formulario para la compra de licencias

Tras proceder a la introducción de los datos referentes a la empresa y el pago[2], así como el número de licencias deseadas, se mostrará un fichero JSON que posee toda la información referente a las licencias adquiridas y que debe ser descargado por el comprador para posteriormente introducirlas dentro de la base de datos de su organización. Un ejemplo de fichero es mostrado en la siguiente figura:

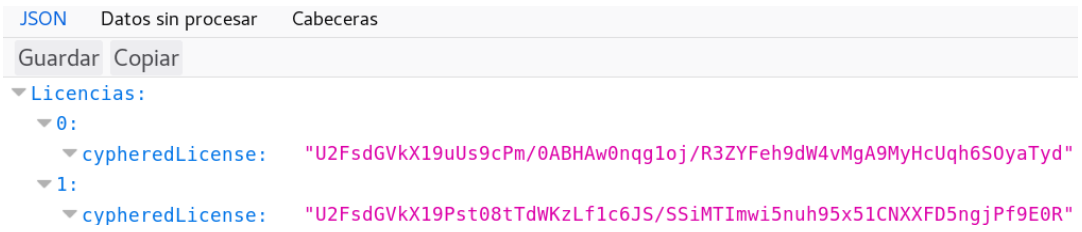


FIGURA 7.9: Ejemplo de fichero JSON con licencias por volumen

[2]El procedimiento de pago seguro es un elemento que no se contempla en el diseño de esta solución, puesto que ya existen mecanismos en el mercado con buenas prestaciones de seguridad (*Paypal*).

Tal y como se puede apreciar en la imagen anterior, al distribuidor nunca se le van a otorgar las licencias en claro, puesto que esto podría dar lugar a la reventa de las mismas y seguirían existiendo los problemas de validación de licencias que quieren evitarse con el diseño de esta solución. Este hecho aporta un mayor grado de confianza, puesto que las licencias originales que se utilizan para acceder a los contenidos *software* unicamente son conocidas por la entidad que actúa como agente de confianza en el sistema (el proveedor de licencias).

Administración de la base de datos

Para finalizar, se incluye una serie de funcionalidades que están enfocadas a la administración de los contenidos almacenados dentro de la base de datos. Para ello, se debe incluir dentro de la lógica de aplicación una sección dedicada a los administradores del sistema, de forma que puedan visualizar el contenido disponible en lo referente a licencias, así como añadir nuevas para reponer el *stock*.

En este punto se pueden añadir diferentes funcionalidades adicionales que permitan a los administradores tener un mayor control sobre el sistema, pero eso queda a elección de las necesidades que posea cada organización. Un ejemplo de las funcionalidades de la página de administración pueden ser las que se muestran a continuación:

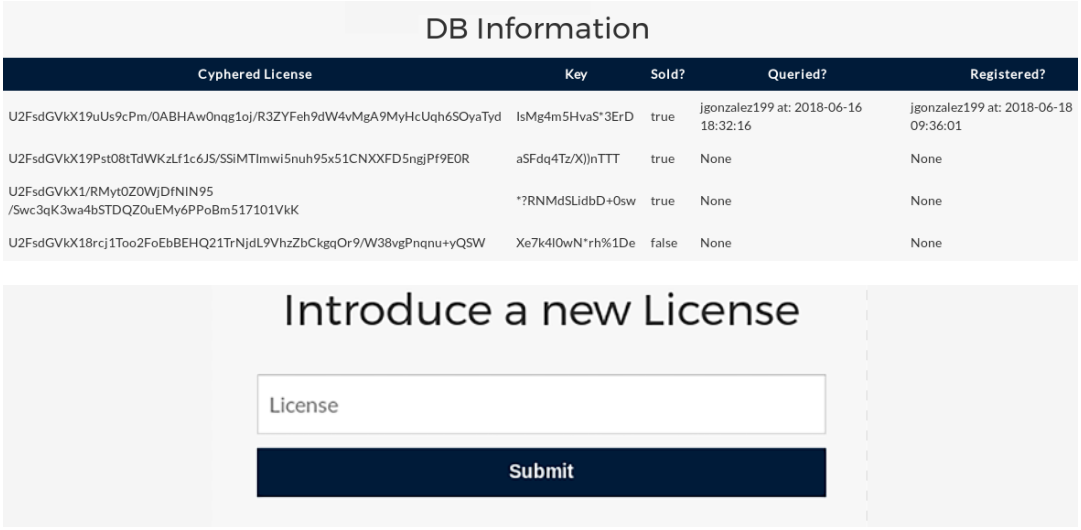


FIGURA 7.10: Ejemplo de funcionalidades para la administración

Esta sección debe ser correctamente securizada, puesto que la información mostrada en la página no puede ser accedida por agentes no autorizados. Para ello, la solución propuesta consiste en la instalación de un servidor web independiente al utilizado por los clientes para acceder a los servicios de compra y verificación de licencias. Este servidor, como es lógico, no puede ser accedido desde el exterior de la red empresarial, por lo que se debe proteger su acceso a través del *firewall* y colocándolo en el área local segura. Además de esto, se deben incluir mecanismos de autenticación que permitan asegurar que solo los empleados con los privilegios requeridos puedan acceder a este servicio.

7.1.2. Control de acceso y autenticación

La infraestructura de red presentada para el proveedor de licencias posee una serie de elementos críticos que deben ser debidamente protegidos para evitar que posibles atacantes puedan vulnerar la seguridad del sistema y comprometer los servicios ofrecidos por la empresa, así como la información utilizada referente a usuarios y licencias *software*. En primer lugar, se va a mostrar un esquemático donde se muestra la arquitectura de red planteada para poder ofrecer un buen nivel de seguridad en el acceso:

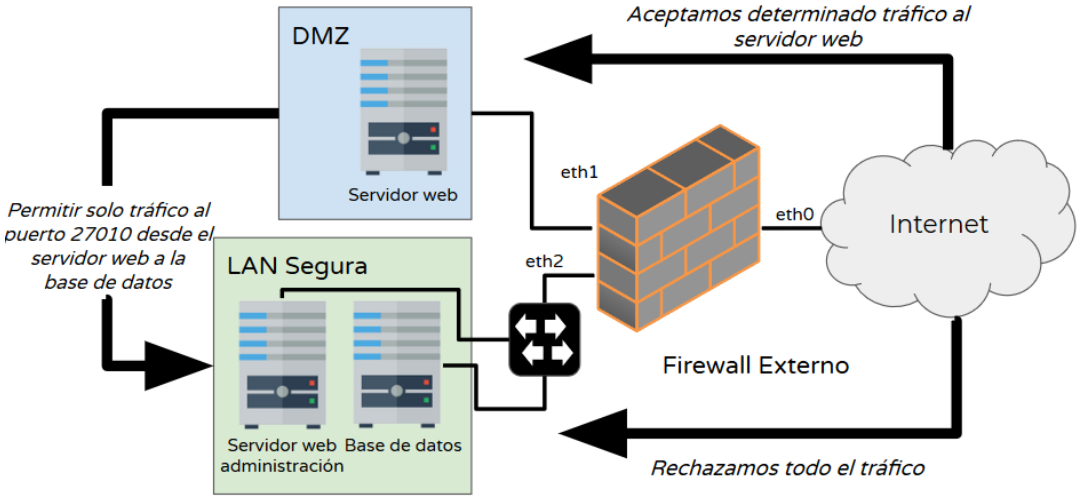


FIGURA 7.11: Arquitectura propuesta para la protección del acceso

Tal y como se adelantó en el análisis de alternativas, una de las arquitecturas a tener en cuenta a la hora de securizar el acceso iba a ser la constituida por un *firewall* y varias interfaces de red. Por medio del cortafuegos, se protege el acceso de los sistemas de información situados en la LAN segura, de forma que no se permita ningún tipo de tráfico externo hacia equipos conectados en dicha interfaz. El servidor web conectado en la DMZ permite el acceso de los clientes a los servicios que se han descrito con anterioridad, mientras que el servidor web de administración que se encuentra situado dentro del área local segura solo puede ser accedido por equipos situados dentro de esa misma red.

Sin embargo, esta no va a ser la única acción considerada. Otra tarea a realizar dentro del modelo consiste en la implementación de autenticación en determinadas rutas y/o equipos para restringir el acceso unicamente a aquellos usuarios que tengan los roles requeridos. Los puntos a proteger dentro del diseño son los siguientes:

- **Servidor web de administración:** Es muy probable que la organización no quiera que todos sus empleados dispongan del acceso a la herramienta que permite gestionar la base de datos y sus contenidos. Por tanto, dentro de la ruta correspondiente a dicho servicio se propone la integración de un sistema de autenticación en el que el propio servidor solicite credenciales a la hora de intentar el acceso a la ruta. A continuación se muestra un ejemplo de control de acceso:

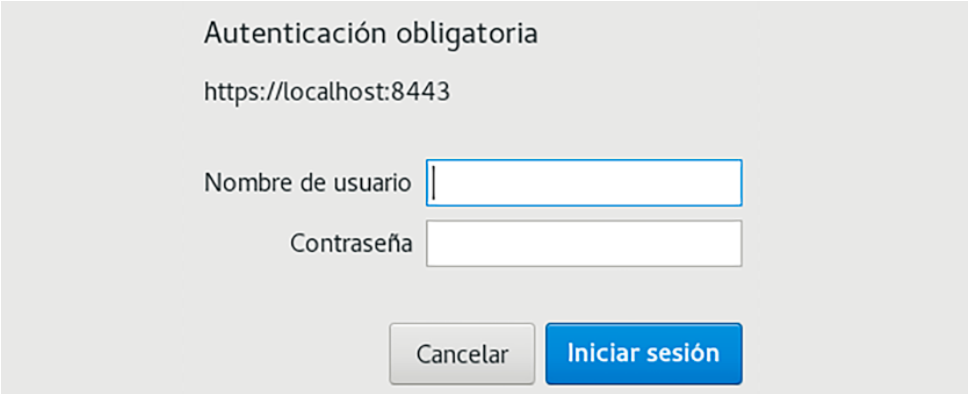


FIGURA 7.12: Autenticación en el acceso a rutas

- **Perfil de los usuarios y servicios de verificación de licencias:** Si un usuario no ha iniciado sesión previamente, no dispondrá de la *cookie* que identifique su sesión. Si se intenta acceder a las rutas correspondientes al perfil o la verificación/registro de licencias y no se envía dicha información en las cabeceras, la petición será rechazada y se redirigirá al usuario a la página web principal.
- **Uso de autenticación en la base de datos:** La conexión a la base de datos debe realizarse siempre de manera autenticada. De esta forma, se puede tener un control más preciso sobre las conexiones realizadas a la misma, y se pueden establecer roles para que cada usuario o entidad disponga de unos privilegios determinados a la hora de realizar las operaciones de lectura y/o escritura. Considerando la arquitectura propuesta, el servidor web utilizado para dar servicio a los clientes debería limitarse a disponer de funciones para lectura y escritura, mientras que el servidor dedicado a la administración puede tener roles con mayores privilegios (*dbOwner, dbAdmin...etc.*).

7.1.3. Base de datos

La base de datos es un sistema crítico para el almacenamiento información, puesto que representa un activo muy importante dentro de la infraestructura planteada. Por tanto, resulta importante definir un buen modelo de datos que permita optimizar en el mayor grado posible el funcionamiento de la misma, y además incluir los mecanismos de seguridad que se precisen para proteger la información contenida en ella. Por ello, se va a dividir este apartado en dos subapartados referentes al almacenamiento y la seguridad que permitan entender de manera más simple el diseño llevado a cabo:

□ Almacenamiento de los datos

Tal y como se propuso en el apartado de análisis de alternativas, la arquitectura de red presentada utiliza como soporte para el almacenamiento de información una base de datos no relacional basada en documentos como es MongoDB. Esta base de datos dispone de un rendimiento elevado y con una flexibilidad adecuada que permite su implementación en casi cualquier escenario.

La arquitectura que se presenta a continuación referencia las diferentes colecciones y campos que requiere el diseño para poder ofrecer los servicios aportados a través de la lógica de aplicación. En este caso concreto, resulta necesario el almacenamiento de información relacionada con los usuarios y licencias. La propuesta incluida en la siguiente imagen se corresponde con un sistema básico que los proveedores requieren para hacer uso del modelo. Sin embargo, estos pueden añadir los campos y colecciones que requieran con total flexibilidad:

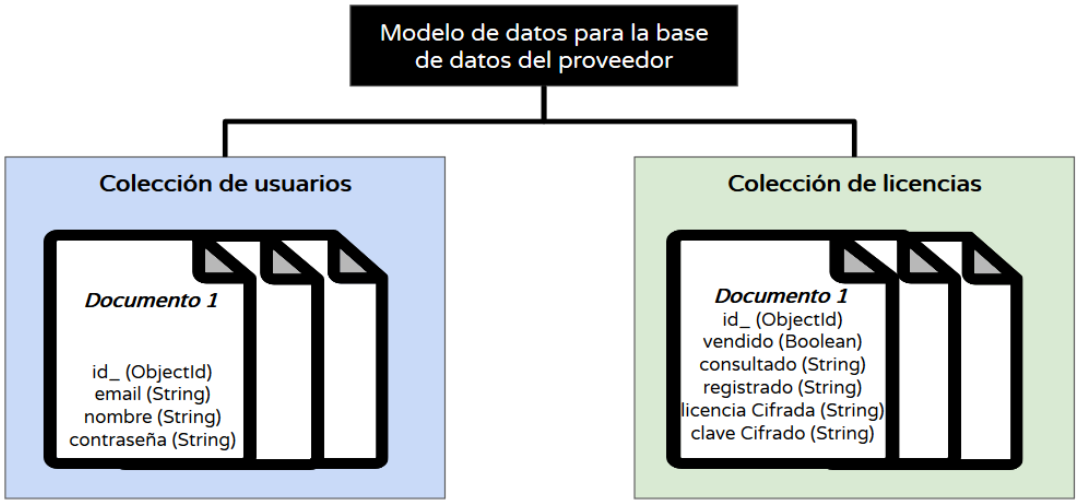


FIGURA 7.13: Propuesta para la base de datos del proveedor

Al tratarse de una base de datos no relacional, no existen uniones entre las diferentes colecciones. Sin embargo, pueden establecerse relaciones por medio de la lógica de aplicación. A modo de clarificar cada uno de los campos utilizados en los documentos y su uso, se incluye a continuación una tabla explicativa:

Campos de la colección de usuarios

Campos	Uso
id_	*ID* del objeto que referencia cada documento
email	*E-mail* del usuario. Usado para el registro y en el inicio de sesión
nombre	Nombre del usuario. Usado para el registro y en el inicio de sesión
contraseña	Contraseña *hasheada*. Creada durante el registro y usada en el inicio de sesión

Campos de la colección de licencias

Campos	Uso
id_	*ID* del objeto que referencia cada documento
vendido	Indica si una licencia ha sido vendida previamente
consultado	Indica que usuario ha consultado una licencia y en que fecha
registered	Indica que usuario ha canjeado una licencia y en que fecha
licencia Cifrada	Licencia cifrada que se vende a los distribuidores

CUADRO 7.2: Descripción de los campos que conforman las colecciones de la base de datos

Una vez vistos los elementos que conforman el modelo de datos, se procede a hablar en el siguiente subapartado sobre las técnicas de securización que se deben implementar para la protección de los datos a nivel de almacenamiento.

☐ Securización de los datos

Aún habiendo utilizado técnicas para la protección del acceso a los sistemas de información, resulta necesario implementar dentro del modelo diferentes técnicas de seguridad que van a proveer un mayor nivel de protección a los datos almacenados. Los puntos que se han tenido en cuenta se detallan a continuación:

- **Contraseñas *hasheadas*:** El almacenamiento de las contraseñas como texto plano resulta muy peligroso, por lo que la solución es optar por realizar un resumen *hash* de las mismas dentro de la lógica de aplicación previamente a su almacenamiento. Por otro lado, un proceso de aleatorización mediante un *salt* va a permitir asegurar de manera más completa la confidencialidad de las contraseñas. El algoritmo a utilizar para ello es ***BCrypt***, puesto que es ampliamente utilizado y su desempeño es adecuado. La imagen mostrada a continuación representa la estructura de una contraseña *hasheada*:

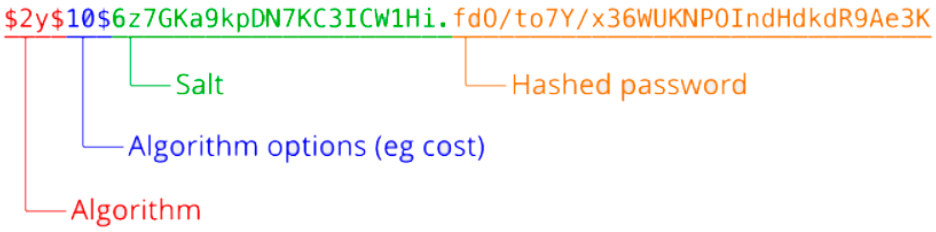


FIGURA 7.14: Estructura de una contraseña *hasheada* con *BCrypt*
Fuente: Stack Overflow

- **Licencias cifradas:** Cuando un administrador introduce nuevas licencias dentro de la base de datos para reponer *stock*, estas son introducidas en texto plano para facilitar dicha labor. Sin embargo, tal y como se ha visto en la Tabla 7.2, dentro de la base de datos se almacena únicamente la licencia ya encriptada. Como clave de cifrado se utilizará una secuencia de caracteres aleatorios para cada licencia. Esta clave debe ser de, por lo menos, 128 *bits* para poder asegurar una buena protección. En lo referente al algoritmo ocurre algo similar, puesto que un nivel de protección adecuado requiere algoritmos de cifrado fuertes, como es el algoritmo simétrico AES[3]. El procedimiento seguido puede observarse en el siguiente diagrama:

[3]La clave generada aleatoriamente sirve tanto para el cifrado como para el descifrado de la licencia.

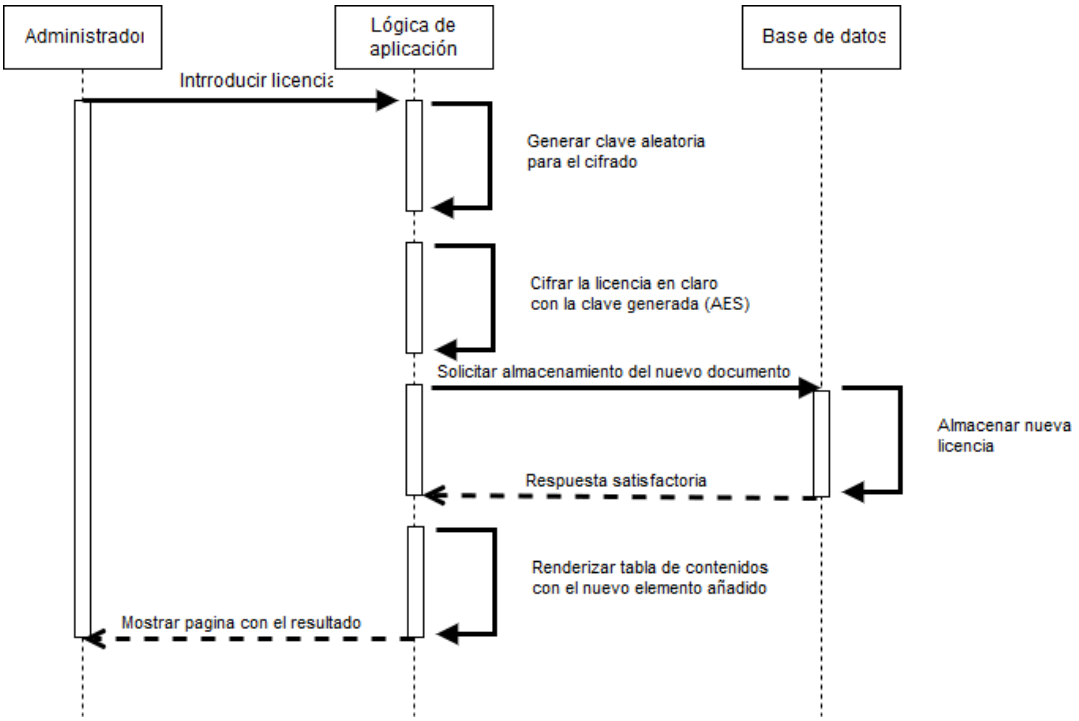


FIGURA 7.15: Proceso de almacenamiento y generación de licencias encriptadas

La gestión de las claves generadas debe realizarse desde la propia lógica de aplicación y estas deben ser almacenadas de forma segura. La opción que se plantea y que se describió con anterioridad en el apartado de análisis de alternativas, es el uso de un HSM instalado dentro del propio servidor que se encargue del almacenamiento seguro de las claves y las entregue al nivel de la lógica de aplicación cuando se requiera. El comportamiento a seguir puede verse en la siguiente imagen:

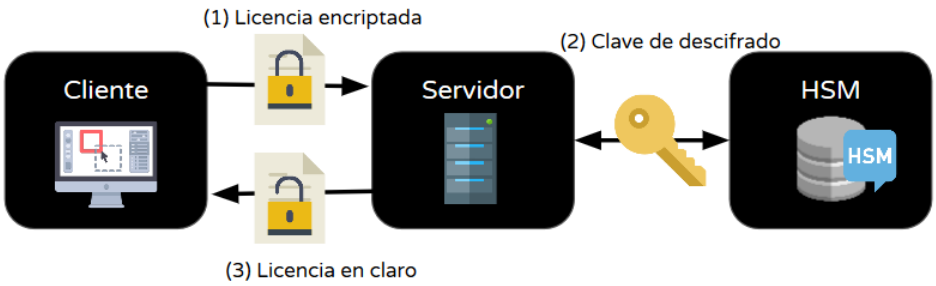


FIGURA 7.16: Proceso de solicitud de la clave de cifrado con un HSM
Fuente: AWS Documentation

7.2. Distribuidor

El punto analizado en esta sección se corresponde con las funcionalidades que debe disponer la empresa encargada de realizar la compra de licencias en grandes volúmenes para después venderlas a usuarios finales a un precio más económico. En primer lugar, se muestra un imagen con un esquemático que representa los elementos fundamentales que incluye esta parte del sistema:

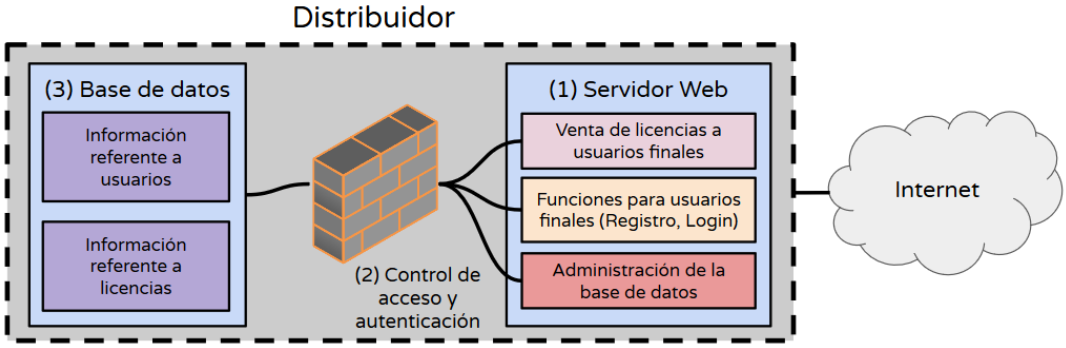


FIGURA 7.17: Propuesta de arquitectura para el distribuidor

Tal y como puede apreciarse en la figura mostrada anteriormente, la arquitectura propuesta no dista mucho de los elementos que posee el proveedor de licencias. En este caso, se puede diferenciar:

- **Servidor web:** Actúa de interfaz entre el distribuidor y los clientes, para que estos últimos puedan acceder a los servicios de compra de licencias.
- **Control de acceso y autenticación:** Mecanismos de seguridad centrados en el control de acceso y la autenticación, de forma que puedan protegerse los datos y servicios gestionados por el distribuidor.
- **Base de datos:** Sistema de almacenamiento que actúa como *backend* y guarda toda la información relacionada con los usuarios registrados en el servicio, así como las licencias por volumen que se han adquirido previamente y que se comercializan a los clientes finales.

En los apartados que se describen a continuación se van a detallar las características y servicios que poseen cada uno de los elementos que componen el distribuidor. Algunas de las funcionalidades provistas son idénticas a las definidas dentro del proveedor de licencias, por lo que estas no se desarrollarán con tanta profundidad:

7.2.1. Servidor web

El servidor web proporciona la interfaz de usuario (*frontend*) para que los clientes puedan acceder a los servicios de compra, así como el *backend* con toda la lógica de negocio que permite que los servicios funcionen correctamente. Si observa detenidamente la Figura 7.17, la lógica de aplicación que provee el servidor web incluye las siguientes funcionalidades básicas:

- **Funciones para el registro y la autenticación de usuarios:** Estas funcionalidades son idénticas a las utilizadas en la parte del proveedor y permiten el registro de los usuarios dentro de la plataforma, así como iniciar sesión para acceder al resto de funciones.
- **Venta de licencias a usuarios finales:** Esta funcionalidad incluye todos los mecanismos requeridos dentro de la lógica de aplicación para ofrecer un servicio de venta de licencias a los usuarios. Esta parte se diseña en concordancia con el proveedor y el gestor de pagos con el objetivo de ofrecer garantías de seguridad al usuario final durante el proceso de compra.
- **Administración de la base de datos:** Al igual que ocurría en el sistema del proveedor de licencias, es necesario un mecanismo que permita gestionar de manera sencilla el contenido dentro de la base de datos. En este caso concretamente, se tratarán las funciones para el almacenamiento estructurado de las licencias por volumen que se han adquirido con anterioridad al proveedor .

A continuación se describen las características y el diseño de cada una de estas funcionalidades de manera independiente en distintos apartados:

Registro de usuarios

La lógica de aplicación utilizada en este punto resulta idéntica a la diseñada previamente en el apartado **Funciones para usuarios finales: Registro**, dentro del proveedor de licencias. El proceso de creación de cuenta se basa en el uso de un formulario donde el cliente introducirá tanto un nombre de usuario, como su correo electrónico y contraseña. La imagen que se muestra a continuación representa una propuesta de página web para el formulario de registro:

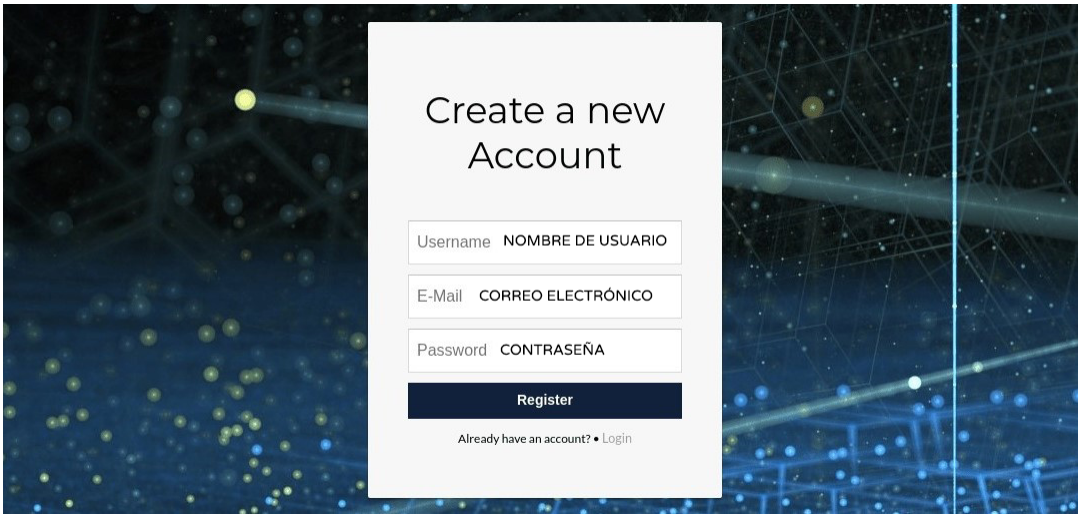


FIGURA 7.18: Propuesta de formulario de registro

Al igual que ocurría en el proveedor, el nombre de usuario y *email* deben ser únicos, y la contraseña se almacenará de manera segura en la DB generando un resumen *hash* de la misma.

Inicio de sesión

El inicio de sesión resulta necesario si se quieren acceder a las funciones de compra de licencias, puesto que el distribuidor requiere tener un control sobre los usuarios que realizan transacciones y compras en su plataforma. El proceso de *login* se realiza a través de un formulario idéntico al mostrado en el apartado **Funciones para usuarios finales: Inicio de sesión**, y que se presenta a través de la siguiente imagen:

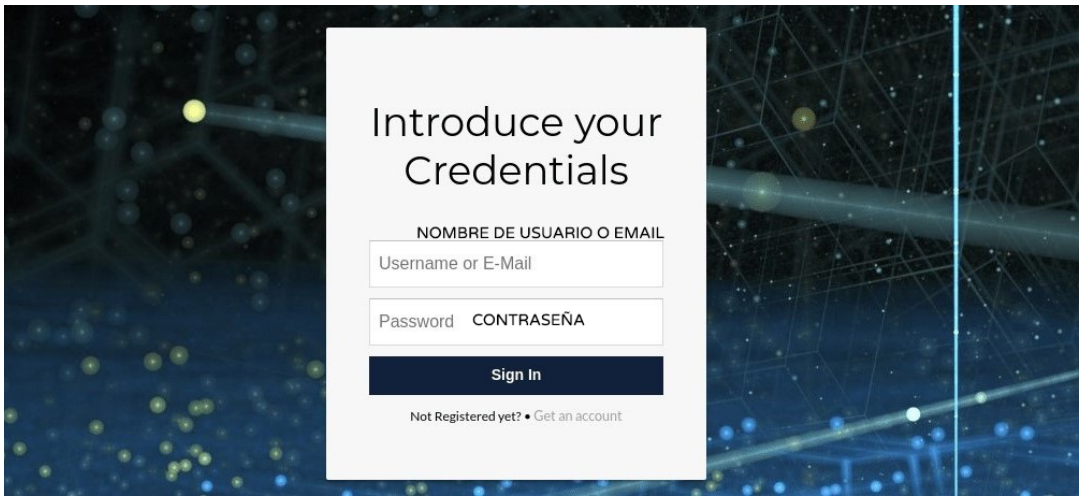


FIGURA 7.19: Propuesta de formulario para el inicio de sesión

El proceso seguido por el sistema para verificar el inicio de sesión es el mismo que el presentado a través del diagrama mostrado en la Figura 7.5.

Venta de licencias a usuarios finales

Este apartado incluye toda la lógica incluida dentro de la aplicación web que va a permitir a los usuarios finales poder adquirir licencias y validarlas frente al proveedor. Tal y como se ha comentado con anterioridad, resulta imprescindible que el usuario se encuentre autenticado para poder acceder a esta funcionalidad.

El sistema propuesto no dista de los métodos que utilizan otras plataformas de venta *online* para comercializar sus productos. En primer lugar, se va a mostrar un formulario donde el cliente debe introducir los datos correspondientes a su compra, así como el método de pago. La diferencia principal es que dentro de la misma pagina se debe incluir un campo rellenado automáticamente, que se corresponde con la licencia que se va a comercializar al usuario. Este código es el que el cliente va utilizar para poder verificar el estado actual de la licencia utilizando la funcionalidad descrita en el apartado de **Funciones para usuarios finales: Verificación y registro de licencias**. En la imagen que se muestra a continuación se presenta una propuesta de formulario de compra:

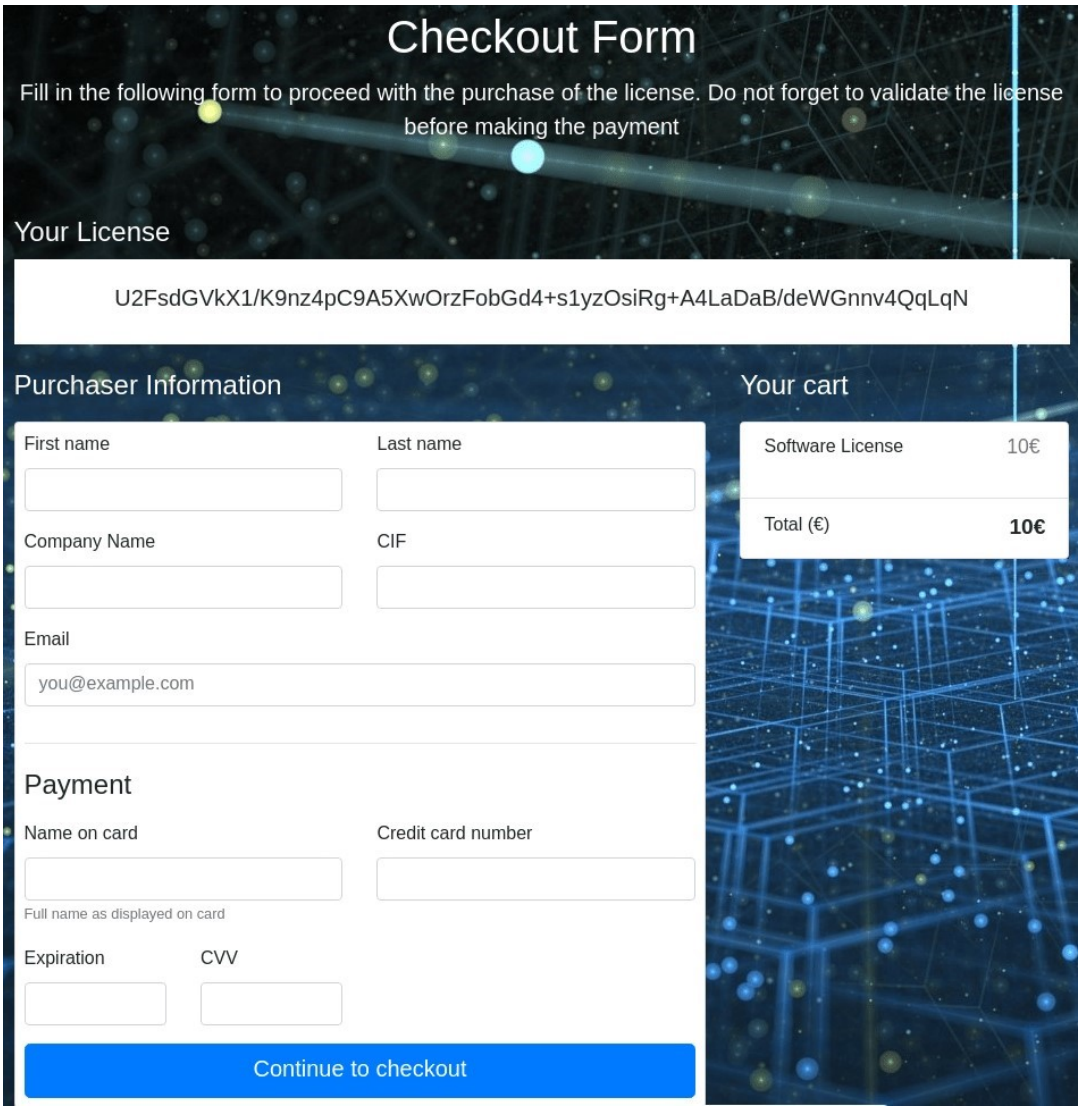


FIGURA 7.20: Propuesta de formulario de compra

Una vez verificado que la licencia es correcta, se procederá con el mecanismo de pago. El distribuidor lanzará una petición de pago al gestor de pagos correspondiente. En caso de que no haya una entrada para dicha licencia, se procederá con el pago y se generará un ticket pseudoaleatorio que permitirá al usuario demostrar que ha realizado la transacción correctamente. Este *ticket* será mostrado de al usuario tras su creación y será también almacenado por el distribuidor para futuras consultas que pueda solicitar el usuario sobre las compras realizadas.

Una vez el usuario dispone de ambos elementos (licencia encriptada y *ticket* de pago), puede proceder al registro final de la licencia utilizando la funcionalidad descrita en el apartado **Funciones para usuarios finales: Verificación y registro de licencias**. En la imagen siguiente se muestra una compra satisfactoria:

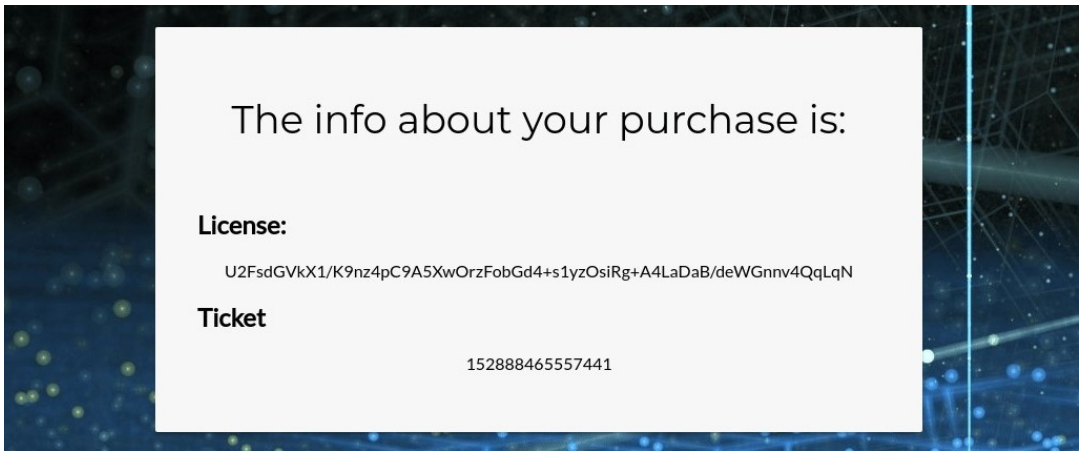


FIGURA 7.21: Compra satisfactoria por parte del cliente

A continuación se representa el proceso completo de compra, incluyendo todas las transacciones entre el usuario final, el distribuidor y el gestor de pagos:

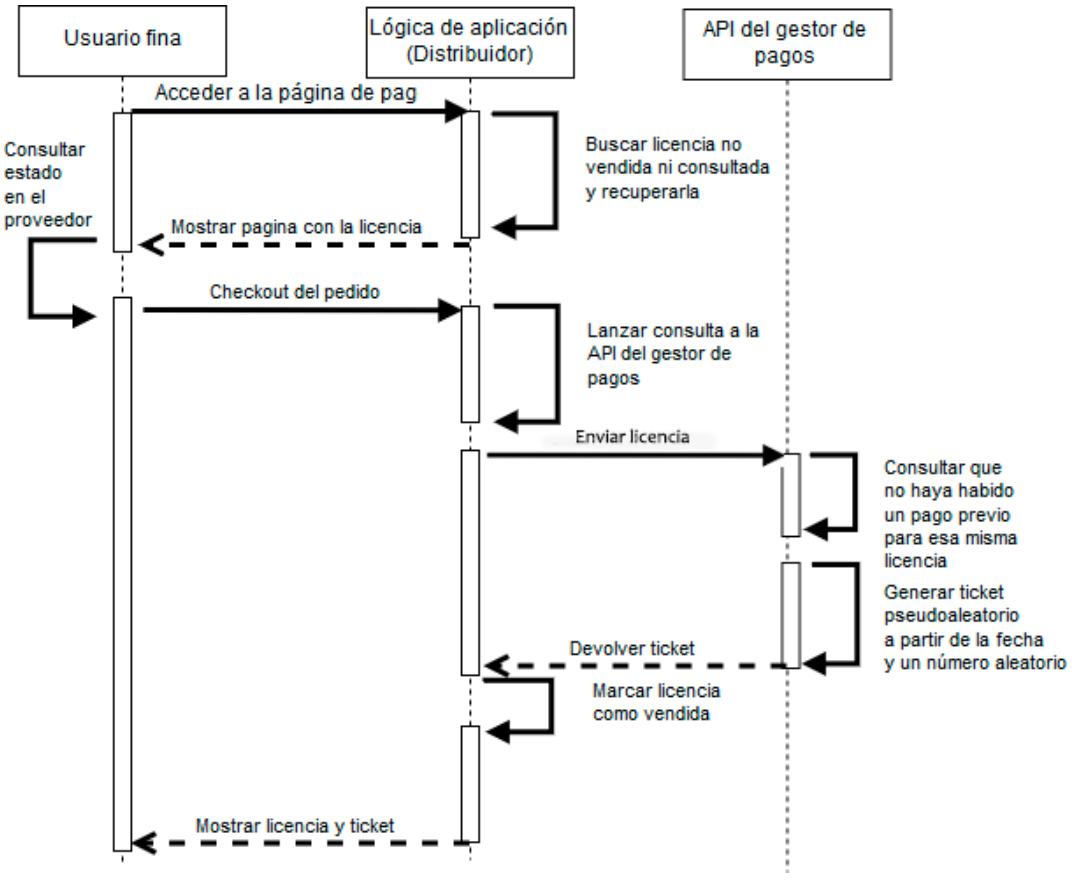


FIGURA 7.22: Diagrama de secuencia para el proceso de compra

Una observación a tener en cuenta en el sistema planteado es que el gestor de pagos no tiene la necesidad de conocer las licencias con las que trabaja tanto el distribuidor como el propio proveedor. Por tanto, dentro del mismo se realizará un procedimiento para la generación del *ticket* que permita identificar la licencia y el pago de manera unívoca sin la necesidad del almacenar esos datos dentro del sistema. Se darán más detalles sobre el procedimiento seguido en el apartado correspondiente al gestor de pagos.

Administración de la base de datos

Para finalizar, al igual que se contemplaba en el proveedor, dentro del distribuidor se incluyen una serie de funcionalidades que están enfocadas a la administración del almacenamiento de datos presente en la arquitectura diseñada.

Esta función tiene una relevancia importante puesto que gracias a ella los administradores del sistema van a ser capaces de añadir las licencias en grandes volúmenes adquiridas de manera automatizada (véase apartado **Compra de licencias por volumen**), así como controlar el contenido disponible dentro de la base de datos. En las siguientes ilustraciones se muestra un concepto de como pueden ser estas funcionalidades dentro de la web de administración:

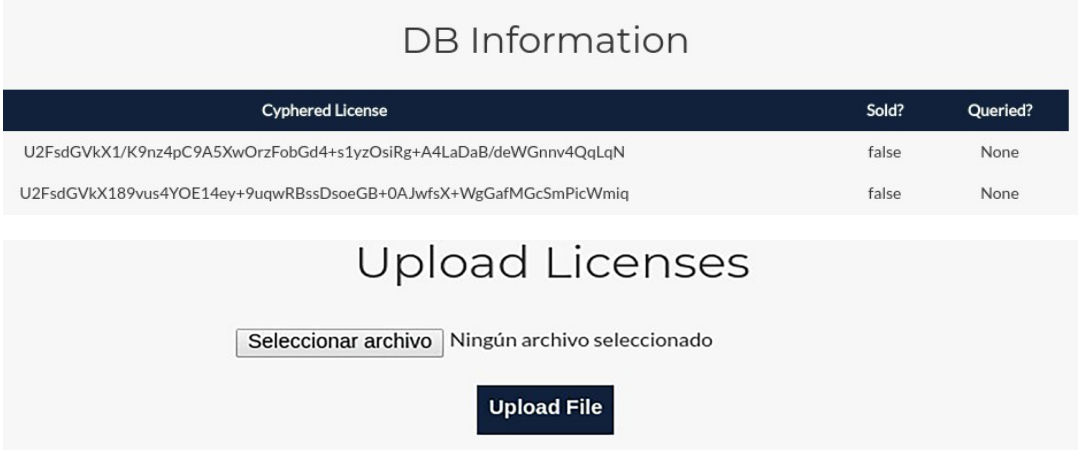


FIGURA 7.23: Propuesta de funcionalidades para administración de la DB

Esta sección debe estar correctamente securizada puesto que las operaciones realizadas en este punto solo pueden ser llevadas a cabo por personal autorizado. Por tanto, tal y como se propuso para el caso del proveedor de licencias, una arquitectura óptima se basa en la instalación de un servidor web independiente dentro del área local segura y que este protegido por el *firewall* y los sistemas de autenticación.

En lo referente al sistema de subida de licencias a la base de datos, se propone el uso de un sistema automatizado que almacene las licencias por volumen adquiridas que se encuentran alojadas dentro del fichero JSON. A través de esta solución, se pueden incluir todas las licencias realizando una única operación y evitando posibles errores. Para ello, se sigue un procedimiento como el mostrado en el siguiente diagrama:

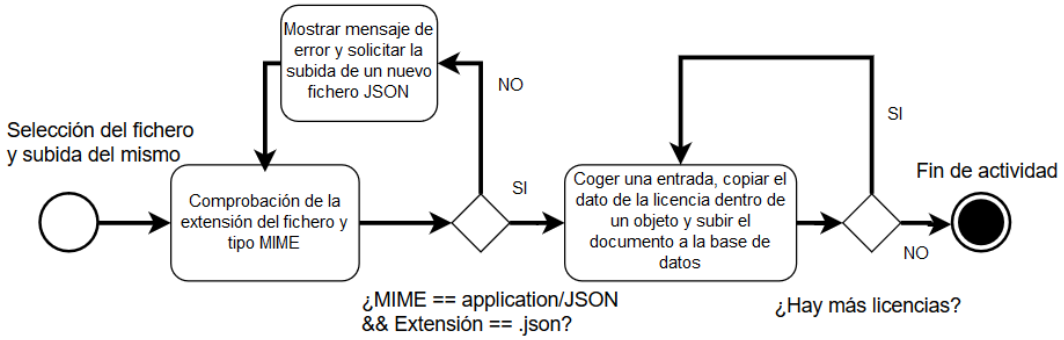


FIGURA 7.24: Diagrama de actividad para el almacenamiento estructurado de las licencias por volumen

Además de la función para añadir las licencias y visualizar la información correspondiente a la base de datos, cada distribuidor puede añadir funcionalidades adicionales dentro de esta sección para poseer un mayor control sobre el sistema. Esta decisión queda a elección de cada organización, puesto que existe total flexibilidad para añadir nuevas funciones para cubrir las necesidades presentes.

7.2.2. Control de acceso y autenticación

Los sistemas de seguridad a implementar en este agente no distan mucho de los que se presentaron previamente para el proveedor de licencias. Estos mecanismos se centran principalmente en asegurar el acceso a los elementos críticos de la infraestructura para evitar posibles ataques y/o filtraciones de información. En primer lugar, se va a mostrar una imagen donde se representa de manera esquemática la propuesta de arquitectura de red para esta organización:

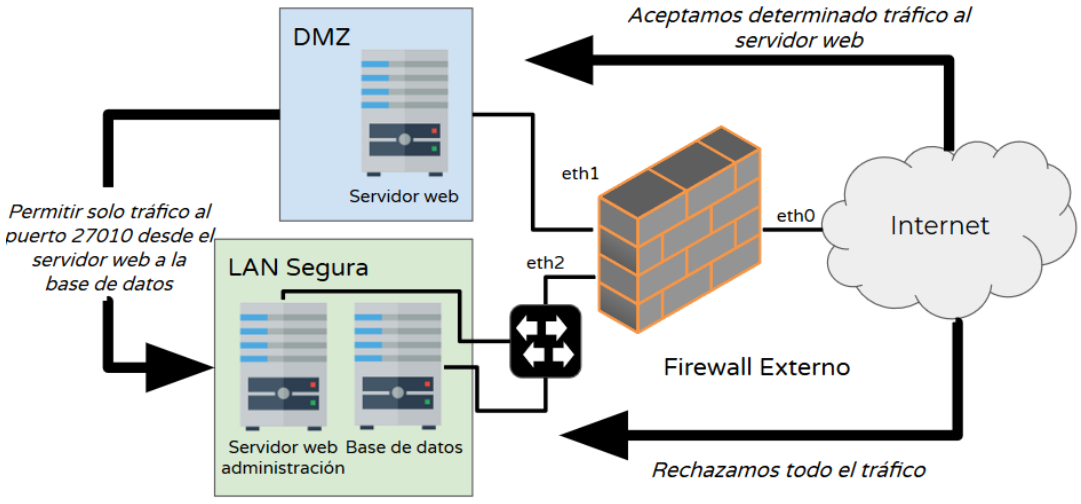


FIGURA 7.25: Arquitectura propuesta para la protección del acceso en el distribuidor

La arquitectura presentada esta basada en la solución adoptada tras la realización del análisis de alternativas. Los usuarios externos van a ser capaces de acceder al servidor web situado en el DMZ para disfrutar de los servicios de compra de licencias. Por otro lado, la base de datos y el servidor web de administración van a situarse dentro de la LAN segura para garantizar que solo los equipos y agentes autorizados puedan acceder a los mismos.

Además de esto, también se deben incluir diferentes mecanismos de autenticación para proteger rutas concretas y restringir el acceso a aquellos usuarios que no tengan los privilegios o roles requeridos. Concretamente, se deben tener en cuenta las siguientes acciones:

- **Servidor web de administración:** Al igual que ocurría en la parte del proveedor, es muy probable que no todas las personas que conforman la organización dispongan de los mismos privilegios a la hora de gestionar los activos de la empresa. Por tanto resulta necesario establecer un nivel de protección adicional dentro del servidor web de administración para asegurar que unicamente los usuarios que dispongan de los privilegios requeridos puedan acceder a las funcionalidades descritas para la gestión de la base de datos. El uso de autenticación en el acceso a las rutas del servidor web pueden garantizar dicha protección (ver Figura 7.12).
- **Perfil de los usuarios y servicio de venta de licencias:** El distribuidor requiere tener un control sobre los usuarios que realizan transacciones dentro de su servicio, así como conocer la identidad de los usuarios en el momento que deseen comprar una licencia. Por tanto, resulta necesario establecer un nivel de protección en el acceso a las rutas que ofrecen los servicios de venta y perfil de usuario. Si se intenta acceder a dichas rutas y no se envía en las cabeceras las *cookies* con la información referente a la sesión del usuario, la petición será rechazada y el cliente será redirigido a la página web principal.
- **Uso de autenticación en la base de datos:** Este mecanismo resulta idéntico al utilizado dentro de la propuesta del proveedor de licencias. Cada uno de los agentes que pueden interactuar con la base de datos deben realizarlo de forma autenticada, de forma que se controlen de manera más exhaustiva las conexiones realizadas, así como las operaciones que cada elemento puede realizar sobre la base de datos. El sistema de roles (*dbOwner, dbAdmin, readWrite, root*...etc.) ayuda a determinar que acciones puede realizar cada uno de los agentes participantes.

7.2.3. Base de datos

Para finalizar con el diseño del modelo en el distribuidor, se va a definir el modelo de datos utilizado para el almacenamiento de la información referente a las licencias comercializadas y los usuarios de la plataforma. Tal y como se propuso en la sección de análisis de alternativas, el tipo de base de datos a utilizar será no relacional basada en documentos, como es MongoDB.

La arquitectura que se va a presentar a continuación referencia las diferentes colecciones y campos que son requeridos dentro del modelo de datos para asegurar un funcionamiento básico del sistema. Sin embargo, los distribuidores pueden añadir campos y colecciones adicionales en función de las necesidades o servicios extra que quieran cubrir, con total flexibilidad. El modelo de datos se presenta en la siguiente imagen:

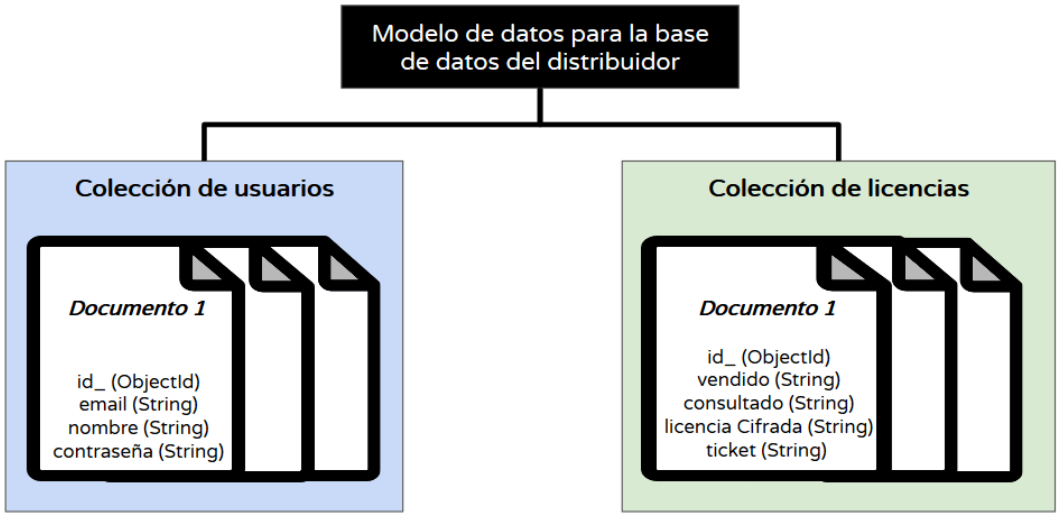


FIGURA 7.26: Propuesta para la base de datos del distribuidor

El modelo presentado es muy similar al utilizado en el proveedor (Figura 7.13). Los campos utilizados y el uso de cada uno de ellos se explican a continuación mediante esta tabla:

Campos de la colección de usuarios

Campos	Uso
id_	*ID* del objeto que referencia cada documento
email	*E-mail* del usuario. Usado para el registro y en el inicio de sesión
nombre	Nombre del usuario. Usado para el registro y en el inicio de sesión
contraseña	Contraseña *hasheada*. Creada durante el registro y usada en el inicio de sesión

Campos de la colección de licencias

Campos	Uso
id_	*ID* del objeto que referencia cada documento
vendido	Indica si una licencia ha sido vendida y a que usuario
consultado	Indica que la licencia encriptada ha sido mostrada a un usuario[4]
licencia Cifrada	Licencia cifrada que se vende a los distribuidores
ticket	Código generado vinculado a una licencia para demostrar el pago de la misma

CUADRO 7.3: Descripción de los campos que conforman las colecciones de la base de datos

[4]Cuando un usuario accede a la sección de compras, en la página web aparece una licencia asignada para su compra (véase Figura 7.20). Por tanto, esa licencia no debe ser mostrada al resto de compradores para evitar un posible problema de duplicidad en las claves vendidas.

En lo referente al almacenamiento seguro, no existen cambios significativos respecto al modelo del proveedor visto con anterioridad. Se van a tener en cuenta las siguientes acciones para proteger los datos durante su almacenamiento:

- **Contraseñas *hasheadas*:** El almacenamiento de las contraseñas se va a realizar por medio de una función resumen que además utilice un proceso de aleatorización mediante un *salt*. El algoritmo a utilizar para ello es nuevamente ***BCrypt***, puesto que su desempeño es adecuado. La estructura de una contraseña *hasheada* por medio de ***BCrypt*** puede verse en la Figura 7.14.
- ***Tickets* cifrados:** Si un agente externo se introduce dentro del sistema y accede a la información contenida en la base de datos, puede registrar los dos parámetros requeridos para validar licencias en la parte del proveedor. En la mayoría de los casos no supondría un problema, puesto que las licencias que no hayan sido vendidas no tendrán un *ticket* vinculado, así como aquellas que dispongan de un *ticket* ya habrán sido registradas por el usuario que las haya comprado.

  Aún así, resulta recomendable proveer un sistema de cifrado que evite que los tickets puedan ser leídos por elementos externos. Para ello, se va a aprovechar el propio algoritmo AES utilizado para cifrar las licencias con el objetivo de generar una versión encriptada de los *tickets*. La clave de cifrado debe ser, al menos, de 128 *bits* para asegurar una encriptación fuerte y su almacenamiento debe ser correctamente gestionado a través del HSM (ver Figura 7.16).

7.3. Gestor de pagos

Esta parte del modelo se corresponde con la entidad que actúa como intermediario entre el proveedor de licencias y el distribuidor, permitiendo el pago de las licencias a los usuarios finales del sistema. En este apartado del documento se van a tratar las diferentes características y funcionalidades que posee este elemento para poder proveer el pago de las licencias, así como facilitar la verificación de los mismos a la parte del proveedor. Como ya se ha hecho para los agentes anteriormente descritos, en primer lugar se va a mostrar una imagen esquemática con las características fundamentales de este punto:

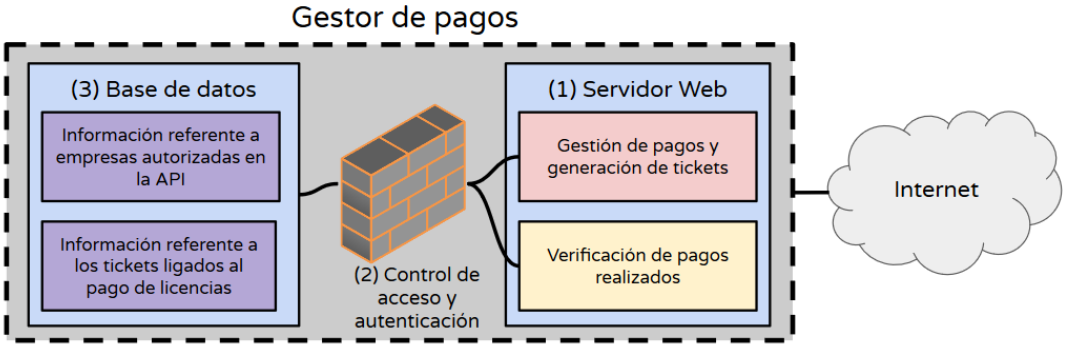


FIGURA 7.27: Propuesta de arquitectura para el gestor de pagos

Tal y como apreciarse en la figura anterior, la mayoría de cambios realizados sobre la arquitectura, en comparación con los modelos vistos anteriormente, se centran en las funcionalidades aportadas por el servidor web y el tipo de datos que se van a almacenar dentro de los sistemas de información. En lo referente a los equipos y mecanismos presentes, el diseño resulta muy similar en los tres puntos descritos hasta el momento. En los subapartados que se incluyen a continuación se describen de manera más extensa las características y servicios presentes en cada uno de los elementos indicados en el modelo:

7.3.1. Servidor web

Una de las diferencias fundamentales respecto al resto de servidores web analizados en los apartados anteriores es que este servidor web va a actuar de forma similar a una *API*. Esto significa que el equipo va a estar continuamente contestando a peticiones concretas realizadas por parte de los servidores web del proveedor y distribuidor, pero sin ofrecer una interfaz de usuario con la que puedan interactuar. Para cada una de las peticiones realizadas se enviarán unos parámetros concretos y se recibirá una respuesta de vuelta con el resultado de la petición. Las dos funciones principales que posee la lógica de aplicación son las siguientes:

- **Gestión de pagos y generación de *tickets*:** Esta funcionalidad esta estrechamente ligada con el distribuidor, puesto que se va a encargar de gestionar los pagos de las licencias por parte de los usuarios finales, así como generar los *tickets* que certifiquen que se ha realizado la compra de las licencias (véase **Venta de licencias a usuarios finales**).
- **Verificación de pagos realizados:** Esta función se vincula directamente con la parte correspondiente al proveedor de licencias, puesto que se encarga de comprobar si para una licencia dada se ha realizado el pago de la misma. En caso de ser así, el proveedor entrega al usuario la licencia final (véase **Funciones para usuarios finales: Verificación y registro de licencias**).

A continuación se describen de manera más extensa cada una de las características y procedimiento seguidos en cada funcionalidad:

Gestión de pagos y generación de *tickets*

Tal y como se ha indicado con anterioridad, la finalidad de esta función es poder ofrecer un sistema que los usuarios puedan utilizar para gestionar los pagos a la hora de realizar la compra de las licencias, así como generar un resguardo que sirva para demostrar que se ha realizado el pago de la licencia correspondiente. Cuando un usuario realice el *checkout* de su compra, dentro del distribuidor se va a lanzar una petición a la *API* del gestor de pagos para que se encargue del pago realizado por el cliente. El procedimiento seguido se incluye en el siguiente diagrama:

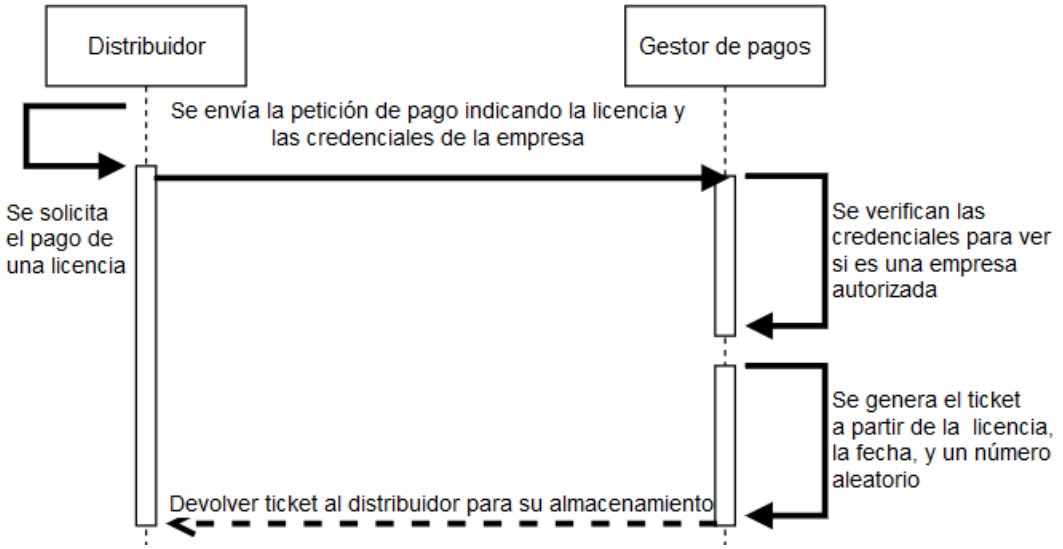


FIGURA 7.28: Diagrama de secuencia para la realización del pago y generación del *ticket*

Si se observa detenidamente la figura anterior, se puede ver como en ningún momento se almacena dentro de la base de datos del gestor de pagos información referente a la licencia que se esta comercializando, sino que simplemente se genera un *ticket* y este se guarda. Este sistema es independiente a la empresa proveedora y a los distribuidores, por lo que no tiene porque manejar los mismos activos en lo referente a información almacenada.

Para seguir manteniendo una relación entre el *ticket* que se va a generar y la licencia que se esta comercializando, se debe desarrollar un procedimiento durante la generación del *ticket* que permita reconocer esa información sin que el gestor de pago almacene de manera explicita la licencia. Por tanto, el diseño de *ticket* que se propone es el que se muestra en la figura siguiente:

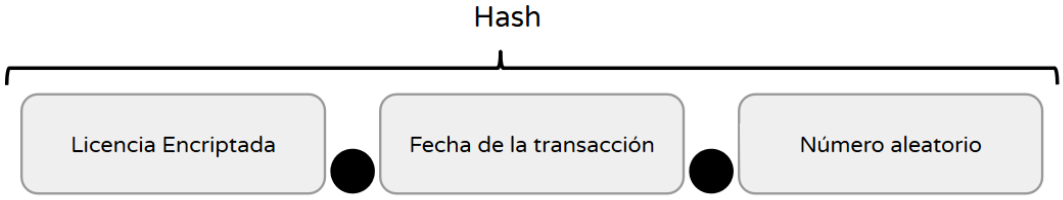


FIGURA 7.29: *Ticket* pseudoaleatorio para la identificación de pagos y licencias

El *ticket* se conforma por los tres elementos mostrados en la imagen anterior, los cuales están separados por un puntos. Para mantener la información referente a la licencia oculta, se procederá a la realización de un resumen *hash* del *string* correspondiente.

Cuando el distribuidor solicite generar un nuevo pago, se comprobará que para la licencia dada no hay ningún pago previo[5], y en caso de que sea correcto, se procederá a la generación del *ticket*.

Verificación de pagos realizados

La segunda funcionalidad presente en la lógica de aplicación consiste en una función que permita a los proveedores verificar el estado del pago de las licencias. Tal y como se puede observar en las funciones de verificación y registro de licencias en el proveedor (véase **Funciones para usuarios finales: Verificación y registro de licencias**), se requiere introducir tanto la licencia encriptada que le proporciona el distribuidor como el *ticket* que certifica el pago de la misma.

Con esos dos parámetros, el servidor web del proveedor debe lanzar una consulta a la *API* del gestor de pagos para conocer la validez de los mismos. En caso de que el usuario haya introducido datos correctos, la *API* devolverá una respuesta satisfactoria y se procederá a la desencriptación de la licencia final. En caso de que no se encuentre el *ticket*, o si este no se encuentra vinculado a la licencia que ha indicado el cliente, se devolverá una respuesta desfavorable al proveedor para que se lo notifique al usuario en cuestión. El procedimiento seguido en el sistema puede verse en el siguiente diagrama:

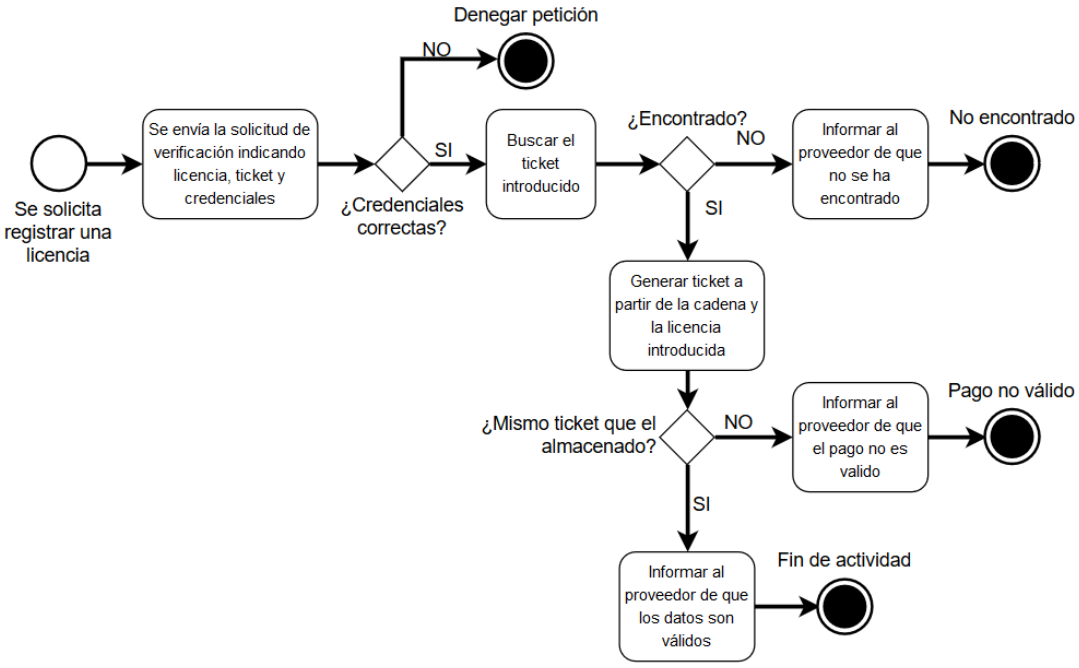


FIGURA 7.30: Diagrama de actividad para la verificación del pago

[5]El procedimiento de generar un *hash* con esa licencia y cada cadena guardada puede suponer un proceso tedioso en caso de que haya un volumen de *tickets* almacenados muy elevado. El hecho de que el usuario pueda comprobar el estado de la licencia previamente al pago permite que este paso no sea imprescindible para asegurar la validez de la misma, aunque puede resultar interesante para evitar que el distribuidor cometa un fraude al intentar registrar dos pagos para una misma licencia.

7.3.2. Control de acceso y autenticación

Las técnicas para el control de acceso y la autenticación son muy similares a las utilizadas en el resto de elementos del modelo. En lo referente al equipamiento y los mecanismos de protección de acceso, se utiliza una arquitectura basada en la inicialmente planteada dentro del análisis de alternativas (usando un *firewall* y varias interfaces de red). El esquema de red utilizado puede observarse en la siguiente figura:

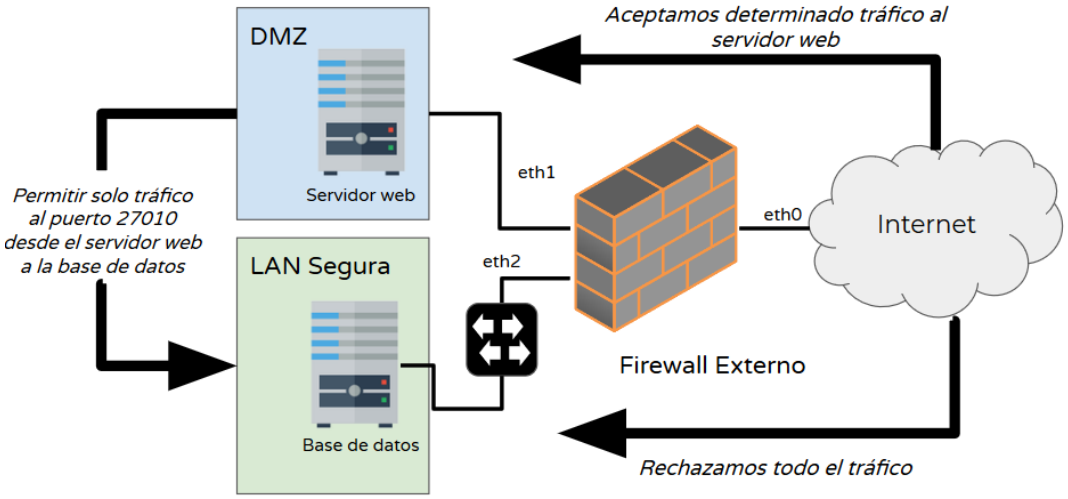


FIGURA 7.31: Arquitectura propuesta para la protección del acceso en el gestor de pagos

En este caso no se contempla el uso de un servidor web para la administración de la base de datos, por lo menos en primera instancia, puesto que la mayoría de los datos van a ser incluidos de manera automatizada por los métodos vistos anteriormente en la lógica de aplicación. Sin embargo, si se considerara interesante disponer de esta funcionalidad, el diseño modular que se ha desarrollado permitiría integrarlo sin demasiada complejidad.

Tal y como se ha visto en los diagramas utilizados para explicar la lógica de aplicación (Figuras 7.28 y 7.30), antes de realizar cualquier operación, la *API* debe verificar que el servidor que realiza la petición es un agente autorizado. Esto implica que deben enviarse en la cabecera de la petición información referente a la identidad de la empresa (nombre y clave secreta[6] utilizada para las comunicaciones entre el proveedor/distribuidor y el gestor de pagos para verificar la identidad de los equipos). El comportamiento seguido se puede observar en el siguiente diagrama:

[6]Esta clave proporciona el acceso a la API del gestor de pago y debe ser correctamente protegida para evitar que agentes externos suplanten la identidad de la empresa y realicen operaciones sobre la misma.

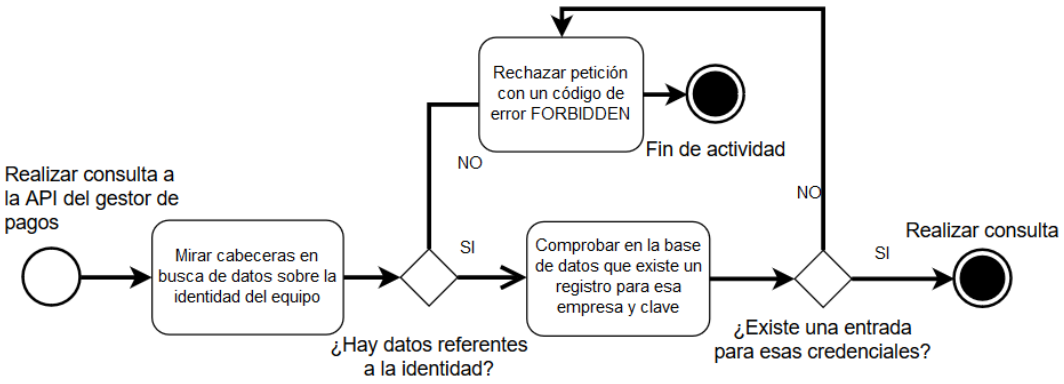


FIGURA 7.32: Diagrama de actividad para la validación de las credenciales de la empresa

Este sistema de seguridad debe ser complementado con el uso de autenticación en la base de datos. Este mecanismo es utilizado en las bases de datos del resto de elementos que conforman el modelo y permite establecer diferentes sistemas de roles y privilegios (*dbOwner, dbAdmin...etc.*) para controlar las operaciones que puede realizar cada equipo o usuario. En el caso del servidor web propuesto, sus privilegios no deberían ir más allá de la lectura y escritura de datos.

7.3.3. Base de datos

Para finalizar con el diseño de este agente, se va a proponer el modelo de datos utilizado para almacenar la información referentes a los *tickets*, así como las credenciales que las empresas autorizadas deben enviar para poder acceder a la *API* y realizar las consultas que requieran. El modelo de base de datos es MongoDB, puesto que se trata de una base de datos no relacional basada en documentos que proporciona las características definidas dentro del análisis de alternativas. El diseño para el almacenamiento de la información se presenta en la siguiente imagen:

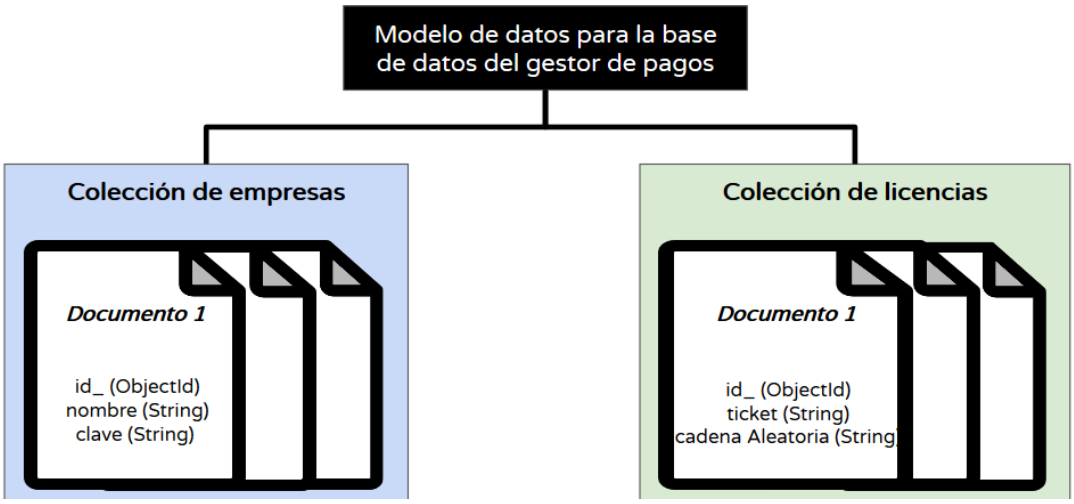


FIGURA 7.33: Propuesta para la base de datos del gestor de pagos

A continuación se incluye una tabla explicativa que define cada uno de los campos utilizados en los documentos, así como el uso que se le da a cada uno de ellos:

Campos de la colección de empresas

Campos	Uso
id_	*ID* del objeto que referencia cada documento
nombre	Nombre de la empresa. Usado para identificar los equipos que lanzan consultas a la *API*
clave	Clave secreta que sirve para validar la identidad de la empresa

Campos de la colección de *tickets*

Campos	Uso
id_	*ID* del objeto que referencia cada documento
ticket	Identificador generado para indicar que una licencia ha sido pagada[7]
cadena Aleatoria	Secuencia utilizada junto con la licencia para generar el *ticket*

CUADRO 7.4: Descripción de los campos que conforman las colecciones de la base de datos

Los campos presentes en el modelo de datos son necesarios para cumplir con las funcionalidades descritas en la lógica de aplicación. Sin embargo, tal como ocurre en el resto de elementos, existe total flexibilidad para que cada organización incluya aquellas colecciones o campos que requieran para ofrecer funcionalidades adicionales. En lo referente al almacenamiento seguro, se deben incluir los siguientes mecanismos con el objetivo de complementar los sistemas de protección en el acceso y evitar que la información se vea comprometida en caso de producirse un ataque:

- **Claves *hasheadas*:** Se va a implementar el algoritmo ***BCrypt*** en las claves secretas utilizas para validar la identidad de la empresa y así evitar un posible robo de las mismas y/o suplantación de identidad. La estructura de una contraseña *hasheada* puede verse en la Figura 7.14.
- ***Tickets* con método de generación propia:** El hecho de utilizar un sistema propietario para almacenar la información del pago y las licencias encriptadas en un elemento único hace que no sea necesario guardar esa información de forma explicita. Esto favorece que se mantenga la privacidad de las licencias ante agentes externos, como en este caso es el propio gestor de pagos[8].

[7] Para ver el procedimiento seguido para generar un *ticket*, consultar el apartado **Gestión de pagos y generación de *tickets***.

[8] Tal y como se ha visto en el procedimiento de generación, el código generado tiene una estructura formada que el propio gestor de pagos no es capaz de descifrar a menos que un elemento externo le indique la licencia vinculada al *ticket*.

7.4. Clientes

Los clientes se constituyen por aquellos usuarios finales que van a utilizar, ya sea de forma directa o indirecta, los diferentes servicios proporcionados por los agentes descritos en apartados anteriores para adquirir las licencias de manera segura. El diseño presentado en este trabajo se basa principalmente en una arquitectura formada por clientes y servidores web que alojan toda la lógica de aplicación referente a la gestión de usuarios y licencias. Por tanto, los clientes unicamente requieren de un equipo con un navegador web instalado para poder realizar el proceso de compra, ya sea un ordenador o dispositivo portátil (móvil, tablet...etc.).

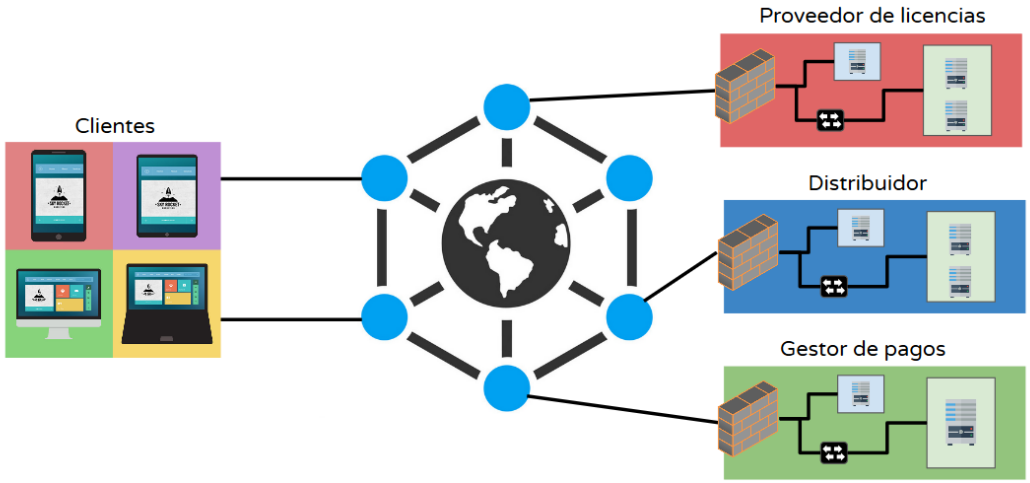


FIGURA 7.34: Infraestructura de red con clientes accediendo a los servicios desde diferentes dispositivos

7.5. Comunicaciones

Este punto representa un elemento importante dentro del diseño planteado, puesto que las transacciones realizadas entre los diferentes agentes que constituyen el modelo son transportadas a través de redes no seguras como Internet. Por tanto, las comunicaciones realizadas deben disponer de unas ciertas garantías de seguridad que aseguren la confidencialidad e integridad de los datos transmitidos.

En primer lugar, y como mecanismo básico de seguridad, se debe forzar el uso del protocolo seguro HTTPS. Este mecanismo, según lo analizado en el apartado de alternativas, se contempla como una solución madura y fiable que se utiliza de manera extendida para las comunicaciones seguras en Internet. El hecho de proporcionar un mecanismo de cifrado con TLS asegura la confidencialidad de la información enviada, mientras que el uso de certificados firmados garantiza que las comunicaciones se realizan con equipos correctamente identificados (evitando así un posible ataque por suplantación de identidad).

Por tanto, para los servidores web accesibles desde el exterior e instalados en las arquitecturas de los diferentes agentes del modelo (proveedor de licencias, distribuidor, gestor de pagos), resulta necesario incluir dentro de la lógica de aplicación un mecanismo que analice las peticiones solicitadas por lo clientes. En el momento que una transacción sea realizada por medio del protocolo HTTP, esta consulta será denegada y se reformulará considerando el protocolo seguro HTTPS. El comportamiento a seguir se describe en el siguiente diagrama:

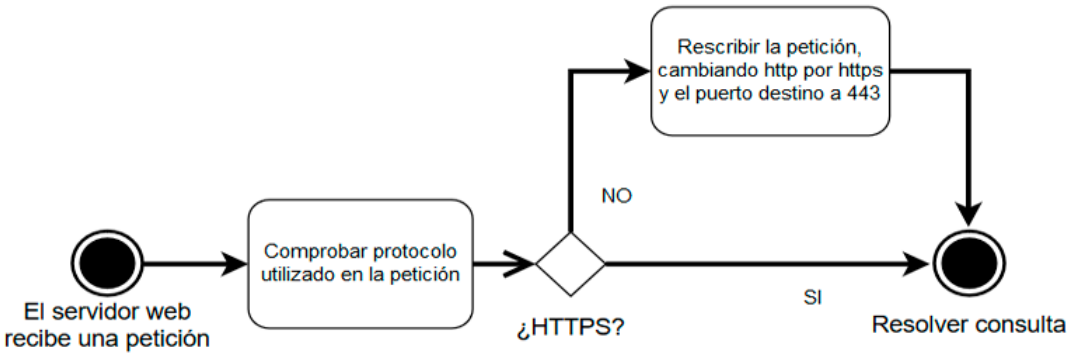


FIGURA 7.35: Diagrama de actividad para la redirección de las consultas con el protocolo HTTPS

Además de esto, dentro de los servidores web debe realizarse otra operación relacionada con la generación de certificados de seguridad. Aunque dentro de una fase de desarrollo se pueden generar certificados autofirmados[9] para comprobar la correcta integración de los mismos con HTTPS, dentro de un entorno de producción resulta necesario adquirir un certificado digital generado por un autoridad certificadora (CA). Tal y como se puede ver en la siguiente imagen, el propio navegador web informará al usuario de que su conexión es segura cuando en el servidor se implemente HTTPS con los procedimientos que se acaban de explicar:

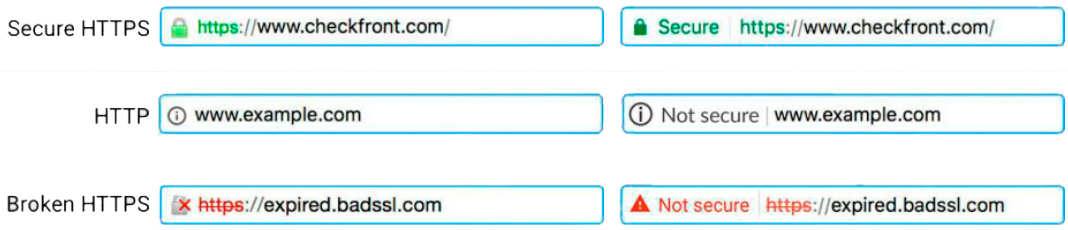


FIGURA 7.36: Verificación del uso de HTTPS. **Fuente: WPXpert**

Para finalizar con esta sección, tal y como se propuso en el apartado de análisis de alternativas, se pueden utilizar de forma complementaria a HTTPS otros mecanismos de seguridad que proporcionen un mayor nivel de protección en las transacciones. En el caso concreto del proveedor de licencias y el distribuidor, resulta necesario mantener información sobre la sesión de los usuarios para poder controlar que los agentes que realizan consultas a los servidores han pasado previamente por un proceso de inicio de sesión.

[9]La herramienta ***OpenSSL*** disponible en Linux permite la creación de la clave privada, así como el CSR y el certificado en si mismo.

Para ello, la solución más adecuada que se plantea en la sección de alternativas consiste en gestionar la propias sesiones en el *middleware* de la aplicación web (ver **Gestión de sesiones en el servidor: El *middleware***). Cuando un usuario inicie sesión dentro de la plataforma del proveedor de licencias o del distribuidor, el servidor almacenará información referente a los datos de la sesión y generará una *cookie* que será devuelta al usuario y que deberá enviar en posteriores peticiones. El procedimiento seguido se puede observar en el siguiente diagrama:

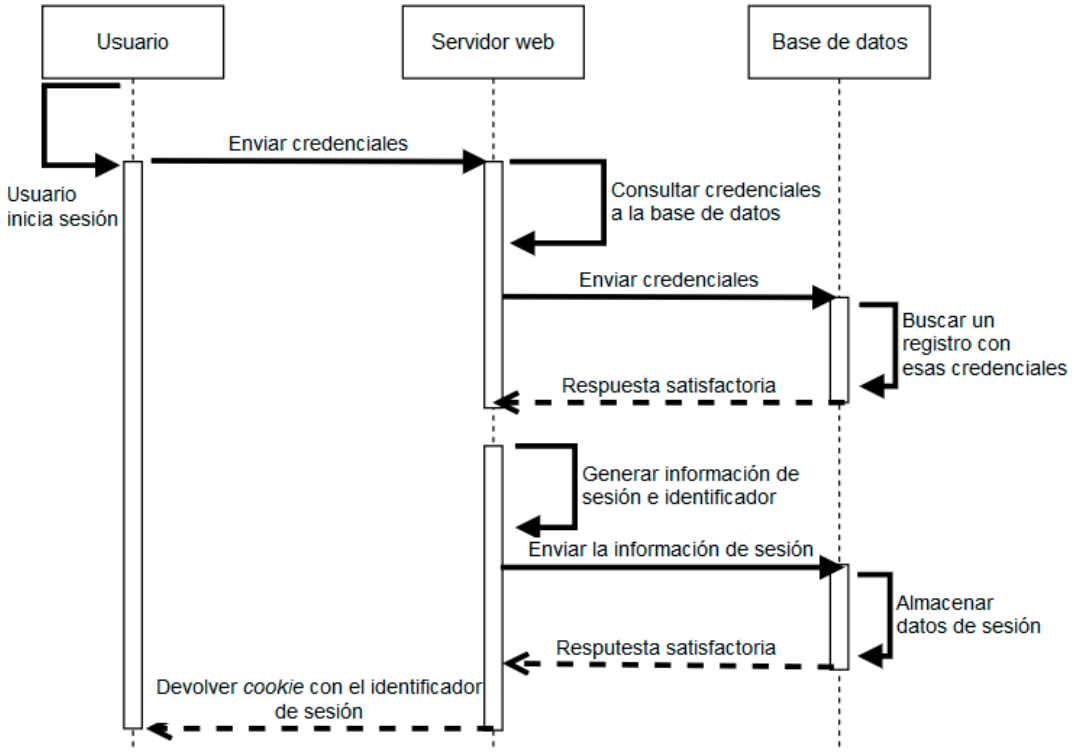


FIGURA 7.37: Diagrama de secuencia para la generación de información de sesión y entrega de la *cookie* al usuario

Las rutas que estén debidamente protegidas y requieran el inicio de sesión para acceder a los contenidos necesitan que el servidor web realice un procedimiento como el mostrado en la siguiente figura:

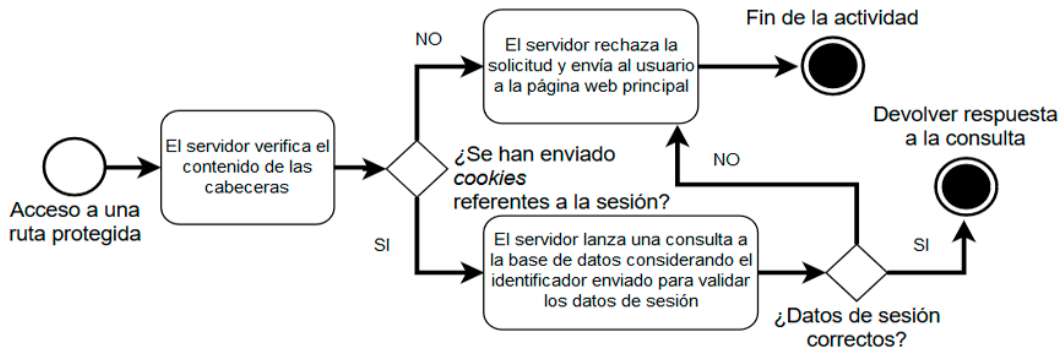


FIGURA 7.38: Diagrama de actividad para validar los datos de sesión

7.6. Funcionamiento general de la solución

Para finalizar con el diseño de la solución, se va a presentar el procedimiento completo para realizar una operación desde que un distribuidor compra licencias en gran volumen hasta que el usuario final registra la licencia definitiva en el proveedor. Los pasos se enumeran a continuación:

- **(1) Compra de licencias en gran volumen:** El distribuidor se conecta al proveedor de licencias y por medio de la interfaz web accede a la sección de compra de licencias. Una vez realizado el pago, el distribuidor recibe un documento JSON donde se incluyen todas las licencias compradas que posteriormente comercializará a los usuarios finales.
- **(2) Introducción de las licencias dentro de la base de datos del proveedor:** Una vez el administrador dispone del fichero con las licencias, debe acceder a la sección de administración de la base de datos alojada en el servidor web interno para procesar el archivo y crear una entrada en la base de datos para cada licencia que se haya adquirido.
- **(3) Verificación de la licencia a adquirir:** Cuando un usuario se conecte a la plataforma del distribuidor y solicite comprar una licencia, en pantalla se le mostrará una código encriptado que debe utilizar para verificar el estado de la licencia que va a comprar. Para ello, debe iniciar sesión igualmente en la plataforma del proveedor de licencias e introducir el código que le ha otorgado el distribuidor.
- **(4) Realización del pago y generación del *ticket*:** Si la licencia ofertada por el distribuidor es válida, el usuario final debe introducir los datos de pago y proceder con el mismo. Una vez realizado el *checkout*, el distribuidor se conectará con el gestor de pagos para validar el pago y generar un *ticket* que certifique la realización del mismo. Ese *ticket* será entregado al usuario final para que pueda registrar su licencia.
- **(5) : Registro final de la licencia:** Una vez el usuario dispone de los dos parámetros requeridos para obtener la licencia final, este se conectará con el proveedor de licencias e introducirá tanto el código referente a la licencia encriptada como el *ticket* generado tras el pago. El proveedor lanzará una petición al gestor de pagos para que valide la existencia de dicho *ticket* y compruebe que el pago realizado se encuentra vinculado a la licencia que ha introducido el cliente. En caso de ser así, se devolverá una respuesta satisfactoria al proveedor que le permitirá desencriptar la licencia y entregársela al usuario final.

En los anexos incluidos al final del documento, se define de manera más extensa el entorno de pruebas que se ha utilizado para simular este sistema dentro de un escenario real y probar que estos mecanismos funcionan correctamente para validar el modelo teórico desarrollado.

8 Descripción de tareas, fases, equipo y planificación

En este apartado del documento se detalla la planificación llevada a cabo para el desarrollo del proyecto. En lo referente a la planificación del trabajo, se ha tenido en cuenta tanto las personas implicadas en el desarrollo y/o documentación, las diferentes tareas y paquetes de trabajo en las que ha sido dividido el proyecto y, finalmente, el reparto temporal estimado para cada una de dichas tareas.

8.1. Equipo de trabajo

En la tabla que se adjunta a continuación se muestra el equipo de trabajo encargado tanto del diseño y desarrollo del modelo presentado en el documento, como de la redacción de la documentación pertinente:

Nivel	Nombre	Organización	Responsabilidad
L1	María Victoria Higuero Aperribay	UPV/EHU	Directora del proyecto
L2	Jose Luis González Calvo	UPV/EHU	Ingeniero Junior

CUADRO 8.1: Equipo de trabajo

donde se entiende L1 y L2 por:

- **Nivel de conocimiento L1:** Ingeniero senior con un alto nivel de conocimiento en el campo de las telecomunicaciones y la ciberseguridad. Su labor principal consistirá en guiar al ingeniero junior, estableciendo las pautas y tareas esenciales que deben llevarse a cabo para lograr un desempeño satisfactorio en el proyecto. Por otro lado, su experiencia puede servir de ayuda para resolver posibles problemáticas que pueden surgir durante el diseño del modelo.
- **Nivel de conocimiento L2:** Ingeniero junior en Tecnología de Telecomunicaciones. Dispone de los conocimientos de seguridad y programación adecuados para el diseño e implementación del modelo presentado en este documento. Su labor es desarrollar las tareas que conforman el proyecto, así como seguir las indicaciones provistas por la directora para obtener resultados acordes a los objetivos que se pretenden conseguir.

8.2. Descripción de tareas y paquetes de trabajo

En este punto se va a proceder a determinar las diferentes tareas y paquetes de trabajo en los que debe ser dividido el diseño e implementación del modelo en este proyecto. Para facilitar la jerarquización del trabajo dentro de diferentes fases y tareas, se va a utilizar una herramienta de gestión de proyectos denominada WBS. Esto va a permitir organizar de manera visual las diferentes actividades que conforman el desarrollo del proyecto. El organigrama se presenta a continuación:

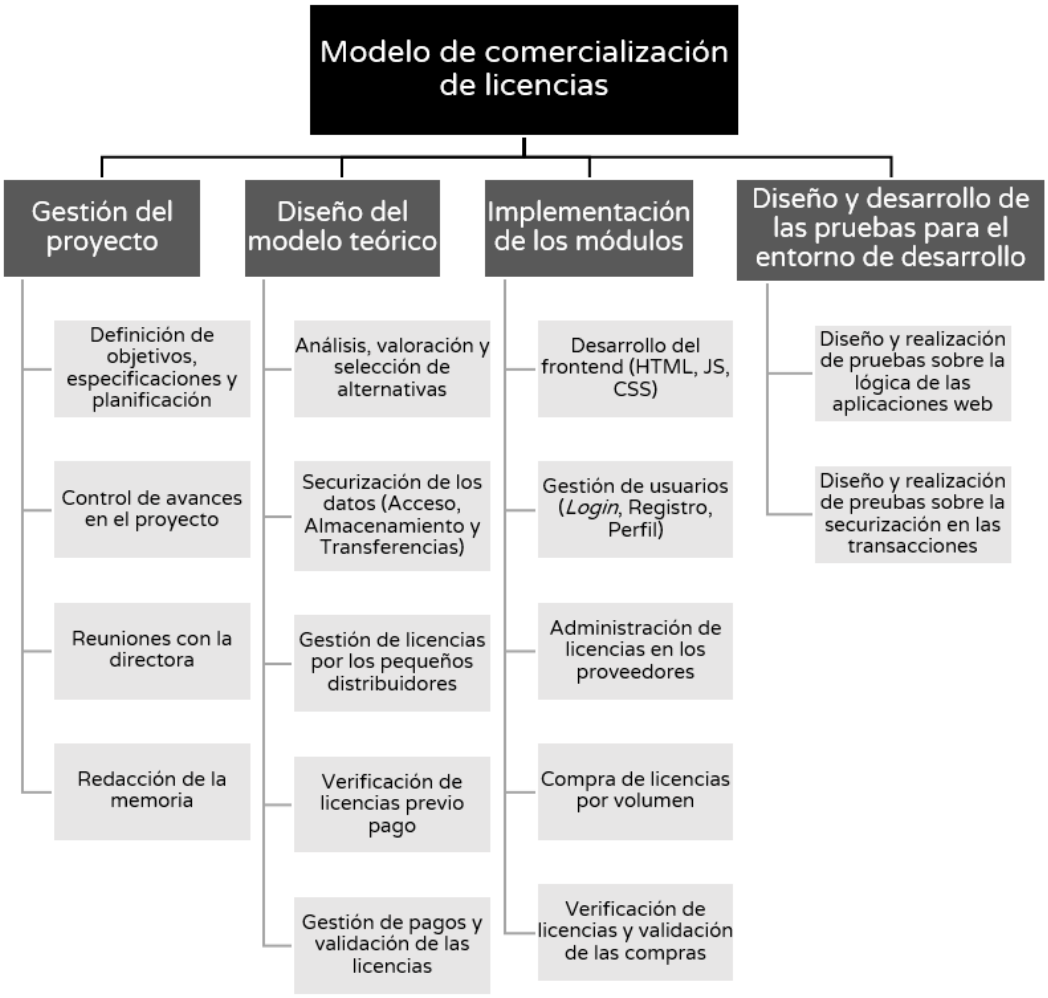


FIGURA 8.1: Paquetes de trabajo del proyecto **(WBS)**

En relación con la ilustración anterior, se incluye una descripción detallada de los paquetes de trabajo y las tareas a realizar en cada uno de ellos:

- **[PT0] Gestión del proyecto:** Este paquete de trabajo incluye una serie de tareas que deben estar presentes en cualquier proyecto y que se centran en la organización y gestión del trabajo a realizar dentro del mismo. En este caso, se pueden destacar estas tareas:

- **(T01) Definición de objetivos, especificaciones y planificación:** Esta tarea consiste en definir los diferentes objetivos que se quieren lograr con el desarrollo del proyecto, así como las características necesarias que va a requerir el modelo para alcanzarlos. También se plantean las pautas de trabajo y la organización necesaria para llevar a cabo dicho desarrollo.
- **(T02) Control de avances en el proyecto:** Esta tarea es ejecutada a lo largo de la duración del proyecto para conocer el progreso que se va alcanzando dentro del desarrollo. Por otro lado, también resulta de utilidad para el establecimiento de medidas correctoras en caso de ser requerido (imprevistos, retrasos...etc.).
- **(T03) Reuniones con la directora:** Esta tarea se complementa con la descrita anteriormente, y supone la realización de diferentes reuniones con la directora del proyecto para controlar el avance del mismo y resolver cualquier problemática o duda que surja durante el desarrollo.
- **(T04) Redacción de la memoria:** Los desarrollos, procedimientos y resultados deben ser correctamente redactados dentro de un documento. A medida que se vaya avanzando en el diseño y desarrollo, se realizará el trabajo de documentación de forma paralela.

- **[PT1] Diseño del modelo teórico:** Este punto constituye una de las fases más importantes del proyecto. Aquí se explicará con gran detalle el conjunto de metodologías y técnicas de seguridad a integrar dentro del modelo. Debido a que el sistema esta constituido por un número de elementos elevado, resulta interesante dividir el trabajo en las siguientes tareas:

  - **(T11) Análisis, valoración y selección de las alternativas:** Una parte importante del proyecto consiste en analizar las diferentes alternativas que pueden utilizarse a la hora de realizar el diseño, valorar las características de cada una de ellas, y seleccionar finalmente aquella que mejor se adapte a las necesidades de cada escenario. Esta parte resulta fundamental para elaborar el análisis de alternativas y el posterior diseño de la solución.
  - **(T12) Securización de los datos:** Esta tarea incluye el diseño de todos los puntos que van a asegurar la protección de los datos almacenados o transmitidos. Para ello, se van a estudiar las diferentes alternativas disponibles y se seleccionarán aquellas más adecuadas según las necesidades que se presenten. Los puntos a tratar están centrados principalmente en la securización en el acceso a los sistemas de información, el almacenamiento estructurado y seguro de los datos, y la transferencia de dicha información de manera fiable a través de redes no seguras como Internet.
  - **(T13) Gestión de licencias por pequeños distribuidores:** Este punto se centra en el diseño de una solución para los pequeños comerciantes que permita definir un método automático para introducir las licencias adquiridas de manera rápida dentro de la base de datos.
  - **(T14) Verificación de licencias previo pago:** Diseño de una solución segura que sea implementada por los pequeños comerciantes en concordancia con los grandes proveedores de licencias para que los clientes puedan verificar el estado de la licencia a adquirir antes de realizar el pago.

- **(T15) Gestión de pagos y validación de las licencias:** Diseño de la entidad encargada de gestionar los pagos y que va a permitir a los proveedores verificar, antes de proveer la licencia al usuario, si dicho usuario ha realizado el pago de la misma.

- **[PT2] Implementación de los módulos:** En este punto se definen los diferentes procedimientos para la programación de los elementos que conforman la maqueta y que permiten comprobar la validez del sistema diseñado. Como tareas referentes a este paquete de trabajo se definen:

  - **(T21) Desarrollo del frontend (HTML, JS, CSS):** Tarea basada en la implementación de las paginas web que los usuarios y administradores van a utilizar para realizar las diferentes operaciones (compra, verificación, control de la base de datos...etc.).
  - **(T22) Gestión de usuarios:** Programación de la lógica de aplicación para ejecutar todas las funciones relacionas con la gestión de usuarios (*login*, registro y acceso al perfil).
  - **(T23) Compra de licencias por volumen:** Desarrollo de la lógica dentro de la aplicación web para que los pequeños distribuidores puedan adquirir grandes volúmenes de licencias para su posterior venta.
  - **(T24) Administración de licencias en los proveedores:** Implementación de la lógica referente a la parte de control de los administradores donde van a ser capaces de controlar, visualizar el contenido de las bases de datos y añadir las licencias adquiridas con comodidad.
  - **(T25) Verificación de licencias y validación de las compras:** Programación de las funciones encargadas de la securización de las licencias, así como de su verificación.

- **[PT3] Diseño y desarrollo de las pruebas para el entorno de desarrollo:** Es necesario añadir una serie de pruebas dentro del proyecto donde se verifique que las características de seguridad concuerdan con lo que se ha definido dentro del modelo y que se cumplen los objetivos fijados. Las pruebas a realizar son las siguientes:

  - **(T31) Diseño y realización de pruebas sobre la lógica de las aplicaciones web:** Se comprobará si el comportamiento de las páginas es el esperado, sin ningún tipo de anomalía en funcionamiento por temas de concurrencia, sincronismo...etc.
  - **(T32) Diseño y realización de pruebas sobre la securización de las transacciones:** Se comprobará que en todos los puntos de la red la información almacenada o transmitida esta correctamente protegida. Por otro lado, también se tendrá en cuenta la protección ofrecida a la hora de restringir el acceso a elementos sensibles como las bases de datos.

8.3. Plan de proyecto y planificación

A continuación se presenta la planificación a seguir dentro del desarrollo del proyecto. Para cada una de las tareas descritas en el apartado anterior se incluye

la duración (sin contar los días festivos), responsable, recursos técnicos/humanos y entregables en caso de haberlos:

[PT0] Gestión de proyecto: Conjunto de tareas relacionadas con la supervisión, organización y gestión del trabajo a realizar dentro del proyecto (**80h**):

Tarea 1: Definición de objetivos, especificaciones y planificación
Tarea enfocada a establecer los objetivos y características fundamentales que deben alcanzarse dentro del proyecto, así como que metodología de trabajo seguir.
Responsable: Jose Luis González Calvo y María Victoria Higuero
Carga de trabajo: 10 horas
Recursos Técnicos: Herramientas ofimáticas (Word, Powerpoint, Editor LaTeX, Gantt Project)
Duración: 14 días
Entregables: Primera aproximación sobre planificación, objetivos y alcance

Tarea 2: Control de avances en el proyecto
Tarea enfocada a realizar un seguimiento en los avances del proyecto para verificar que se cumplan los objetivos propuestos.
Responsable: Jose Luis González Calvo y María Victoria Higuero
Carga de trabajo: 20 horas
Recursos Técnicos: Herramientas ofimáticas (Word) y VCS (Git)
Duración: 221 días
Entregables: -

Tarea 3: Reuniones con la directora
Tarea enfocada a realizar reuniones periódicas con la directora del proyecto con el objetivo de que controle el avance del proyecto y resuelva cualquier problemática o duda que pueda surgir.
Responsable: Jose Luis González Calvo y María Victoria Higuero
Carga de trabajo: 10 horas
Recursos Técnicos: Herramientas ofimáticas (Word, Powerpoint)
Duración: 235 días
Entregables: -

Tarea 4: Redacción de la memoria
Redacción del documento correspondiente a la memoria final del proyecto, así como su correspondiente revisión.
Responsable: Jose Luis González Calvo y María Victoria Higuero
Carga de trabajo: 40 horas
Recursos Técnicos: Internet, Herramientas ofimáticas (Word, Powerpoint), Editor de fotografía (Adobe Photoshop), Editor LaTeX
Duración: 235 días
Entregables: Memoria final del proyecto

[PT1] Diseño del modelo teórico: Conjunto de tareas enfocadas al diseño de una solución que incluya diferentes metodologías y técnicas de seguridad para definir un modelo de comercialización de licencias fiable para los clientes y eficiente para los proveedores/distribuidores (**150h**):

Tarea 1: Análisis, valoración y selección de alternativas
Estudio, análisis y valoración de las diferentes alternativas que pueden llegar a utilizarse dentro del diseño, considerando cuales resultan más adecuada en función de los escenarios planteados.
Responsable: Jose Luis González Calvo
Carga de trabajo: 50 horas
Recursos Técnicos: Internet, Artículos de investigación, Editor LaTeX
Duración: 76 días
Entregables: Análisis de alternativas

Tarea 2: Securización de los datos
Diseño de los puntos que asegurar la protección del sistema. Estos son la securización del acceso a los sistemas de información, el almacenamiento seguro y estructurado y la transferencia fiable de datos a través de Internet.
Responsable: Jose Luis González Calvo
Carga de trabajo: 20 horas
Recursos Técnicos: Internet, Artículos de investigación, Editor LaTeX
Duración: 20 días
Entregables: -

Tarea 3: Gestión de licencias por pequeños comerciantes
Tarea enfocada en el diseño de una solución para los pequeños comerciantes que permita definir un método automático para introducir las licencias por volumen adquiridas de manera rápida dentro de la base de datos.
Responsable: Jose Luis González Calvo
Carga de trabajo: 20 horas
Recursos Técnicos: Internet, Artículos de investigación, Editor LaTeX
Duración: 30 días
Entregables: -

Tarea 4: Verificación de licencias previo pago
Diseño de una solución segura que se implemente en concordancia entre pequeños distribuidores y grandes proveedores para que los clientes puedan verificar el estado de la licencia sin haber realizado previamente el pago.
Responsable: Jose Luis González Calvo
Carga de trabajo: 30 horas
Recursos Técnicos: Internet, Artículos de investigación, Editor LaTeX
Duración: 30 días
Entregables: -

Tarea 5: Gestión de pagos y validación de licencias
Tarea enfocada en el diseño de las características referentes a la entidad encargada de gestionar los pagos y que va a permitir a los proveedores conocer el estado del pago de una licencia previamente a entregarsela al usuario.
Responsable: Jose Luis González Calvo
Carga de trabajo: 30 horas
Recursos Técnicos: Internet, Artículos de investigación, Editor LaTeX
Duración: 44 días
Entregables: Descripción de la solución

[PT2] Implementación de los módulos: Tareas referentes a la programación de los elementos que conforman la maqueta y permiten comprobar la validez del sistema diseñado (**120h**):

Tarea 1: Desarrollo del *frontend* (HTML, JS, CSS)
Tarea basada en la implementación de las páginas web que los usuarios y administradores van a utilizar para acceder a los servicios y realizar operaciones (compra, verificación de licencias, control de la base de datos...etc.).
Responsable: Jose Luis González Calvo
Carga de trabajo: 15 horas
Recursos Técnicos: Internet, Libros de consulta, Visual Studio Code
Duración: 25 días
Entregables: -

Tarea 2: Gestión de usuarios
Programación de la lógica de aplicación que proporcione todas las funcionalidades relacionadas con la gestión de usuarios (*login*, registro y acceso al perfil).
Responsable: Jose Luis González Calvo
Carga de trabajo: 20 horas
Recursos Técnicos: Internet, Libros de consulta, Visual Studio Code
Duración: 21 días
Entregables: -

Tarea 3: Compra de licencias por volumen
Desarrollo de la lógica de aplicación que permita a los pequeños distribuidores poder adquirir grandes volúmenes de licencias para su posterior venta.
Responsable: Jose Luis González Calvo
Carga de trabajo: 25 horas
Recursos Técnicos: Internet, Libros de consulta, Visual Studio Code
Duración: 17 días
Entregables: -

Tarea 4: Administración de licencias en los proveedores
Implementación de la lógica referente a la parte de control de los administradores, donde van a ser capaces de visualizar el contenido de las bases de datos y añadir las licencias adquiridas con comodidad.
Responsable: Jose Luis González Calvo
Carga de trabajo: 30 horas
Recursos Técnicos: Internet, Libros de consulta, Visual Studio Code
Duración: 21 días
Entregables: -

Tarea 5: Verificación de licencias y validación de las compras
Tarea enfocada en la programación de las funciones encargadas de securización de las licencias, así como de su verificación.
Responsable: Jose Luis González Calvo
Carga de trabajo: 30 horas
Recursos Técnicos: Internet, Artículos de investigación, Editor LaTeX
Duración: 21 días
Entregables: Maqueta funcional del modelo

[PT3] Diseño y desarrollo de las pruebas para el entorno de desarrollo: Conjunto de pruebas a realizar para asegurar la conformidad del modelo diseñado en lo referente a los objetivos propuestos (**30h**):

Tarea 1: Diseño y realización de pruebas sobre la lógica de las aplicaciones web
Tarea enfocada a comprobar que el comportamiento de las páginas web es el esperado, sin anomalías en rendimiento, concurrencia, sincronismo...etc.
Responsable: Jose Luis González Calvo
Carga de trabajo: 15 horas
Recursos Técnicos: Visual Studio Code, Navegador web
Duración: 4 días
Entregables: -

Tarea 2: Diseño y realización de pruebas sobre la securización de las transacciones
Tarea centrada en comprobar que la información almacenada o transmitida se encuentra correctamente securizada en cualquier punto de la red que conforma el modelo.
Responsable: Jose Luis González Calvo
Carga de trabajo: 15 horas
Recursos Técnicos: Visual Studio Code, Navegador web, Generador de peticiones (Postman)
Duración: 7 días
Entregables: Plan de pruebas y resultados

8.4. Diagrama de Gantt

A continuación se ha incluido un cronograma donde se muestra de manera más intuitiva las diferentes tareas que conforman el proyecto, así como el tiempo dedicado a las mismas. Esto permite organizar de manera más visual la planificación del proyecto.

El proyecto finalizará el día 29 de junio de 2018 con la entrega de la memoria final en la plataforma ADDI. La tabla que se muestra a continuación muestra las fechas inicio y fin de proyecto, así como la duración y la carga de trabajo del mismo:

Fecha de inicio	Fecha de finalización	Duración	Carga de trabajo
28 de octubre de 2017	29 de junio de 2018	235 días	380 horas

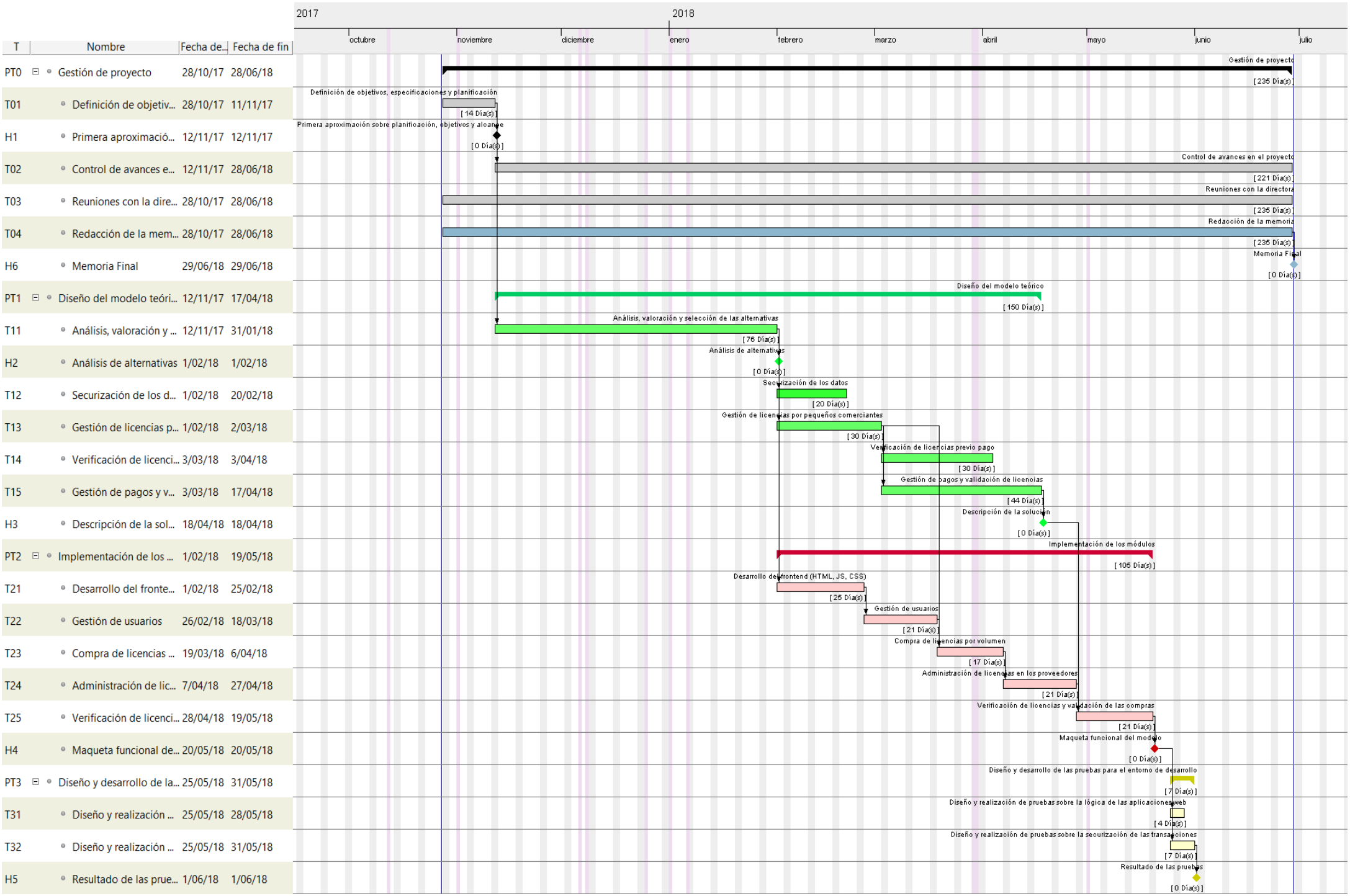
CUADRO 8.2: Fechas de inicio, fin y duración del proyecto

En lo referente a los hitos del proyectos, se han definidos los que se muestran en la siguiente tabla:

Código de hito	Descripción
H1	Primera aproximación sobre planificación, objetivos y alcance
H2	Análisis de alternativas
H3	Descripción de la solución
H4	Maqueta funcional del modelo
H5	Plan de pruebas y resultados
H6	Memoria Final

CUADRO 8.3: Hitos del proyecto

Para finalizar con la planificación, se adjunta el diagrama de Gantt correspondiente en la siguiente página:


2017
2018
octubre
noviembre
diciembre
enero
febrero
marzo
abril
mayo
junio
julio
T
Nombre
Fecha de...
Fecha de fin
PT0 Gestión de proyecto 28/10/17 28/06/18
T01 Definición de objetiv... 28/10/17 11/11/17
H1 Primera aproximació... 12/11/17 12/11/17
T02 Control de avances e... 12/11/17 28/06/18
T03 Reuniones con la dire... 28/10/17 28/06/18
T04 Redacción de la mem... 28/10/17 28/06/18
H6 Memoria Final 29/06/18 29/06/18
PT1 Diseño del modelo teóri... 12/11/17 17/04/18
T11 Análisis, valoración y ... 12/11/17 31/01/18
H2 Análisis de alternativas 1/02/18 1/02/18
T12 Securización de los d... 1/02/18 20/02/18
T13 Gestión de licencias p... 1/02/18 2/03/18
T14 Verificación de licenci... 3/03/18 3/04/18
T15 Gestión de pagos y v... 3/03/18 17/04/18
H3 Descripción de la sol... 18/04/18 18/04/18
PT2 Implementación de los ... 1/02/18 19/05/18
T21 Desarrollo del fronte... 1/02/18 25/02/18
T22 Gestión de usuarios 26/02/18 18/03/18
T23 Compra de licencias ... 19/03/18 6/04/18
T24 Administración de lic... 7/04/18 27/04/18
T25 Verificación de licenci... 28/04/18 19/05/18
H4 Maqueta funcional de... 20/05/18 20/05/18
PT3 Diseño y desarrollo de la... 25/05/18 31/05/18
T31 Diseño y realización ... 25/05/18 28/05/18
T32 Diseño y realización ... 25/05/18 31/05/18
H5 Resultado de las prue... 1/06/18 1/06/18
Gestión de proyecto
[235 Día(s)]
Definición de objetivos, especificaciones y planificación
[14 Día(s)]
Primera aproximación sobre planificación, objetivos y alcance
[0 Día(s)]
Control de avances en el proyecto
[221 Día(s)]
Reuniones con la directora
[235 Día(s)]
Redacción de la memoria
[235 Día(s)]
Memoria Final
[0 Día(s)]
Diseño del modelo teórico
[150 Día(s)]
Análisis, valoración y selección de las alternativas
[76 Día(s)]
Análisis de alternativas
[0 Día(s)]
Securización de los datos
[20 Día(s)]
Gestión de licencias por pequeños comerciantes
[30 Día(s)]
Verificación de licencias previo pago
[30 Día(s)]
Gestión de pagos y validación de licencias
[44 Día(s)]
Descripción de la solución
[0 Día(s)]
Implementación de los módulos
[105 Día(s)]
Desarrollo del frontend (HTML, JS, CSS)
[25 Día(s)]
Gestión de usuarios
[21 Día(s)]
Compra de licencias por volumen
[17 Día(s)]
Administración de licencias en los proveedores
[21 Día(s)]
Verificación de licencias y validación de las compras
[21 Día(s)]
Maqueta funcional del modelo
[0 Día(s)]
Diseño y desarrollo de las pruebas para el entorno de desarrollo
[7 Día(s)]
Diseño y realización de pruebas sobre la lógica de las aplicaciones web
[4 Día(s)]
Diseño y realización de pruebas sobre la securización de las transacciones
[7 Día(s)]
Resultado de las pruebas
[0 Día(s)]

9 Resumen de costes

El desarrollo de este proyecto ha supuesto una serie de costes, los cuales se ven reflejados en el siguiente resumen:

Recursos humanos

Concepto	Coste horario (€/hora)	Número de horas	Coste (€)
Ingeniero junior (Diseño e implementación)	15	330	5025
Ingeniera senior (Supervisión)	40	50	2000
TOTAL			**7025**

Amortizaciones

Concepto	Coste de adquisición (€)	Vida útil	Coste total (€/año)
Ordenador portátil	1000	5 años	200
PC de sobremesa	400	5 años	80
Licencia Microsoft Office	150	1 año	150
TOTAL			**430**

Subcontrataciones[1]

Concepto	Coste horario (€/hora)	Número de horas	Coste (€)
-	-	-	-
TOTAL			**0**

[1]Este proyecto ha sido desarrollado en su totalidad por los miembros que conforman el equipo de trabajo (véase **Equipo de trabajo**). Por tanto, los costes por contrataciones externas son inexistentes.

Gastos

Concepto	Coste (€)
Facturas de Internet	450
Facturas de electricidad	540
TOTAL	**990**

Costes totales del proyecto

Concepto	Coste (€)
Recursos humanos	7025
Amortizaciones	430
Subcontrataciones	0
Gastos	990
Subtotal	8445
Imprevistos (+10 %)	844.5
TOTAL	**9289.5**

CUADRO 9.1: Resumen de costes del proyecto

9.1. Análisis y costes totales del proyecto

El desarrollo de este trabajo supone un **coste total** que asciende a la cifra de **9289.5 €(IVA incluido)**. La mayoría de los recursos económicos son utilizados para hacer frente a los gastos en recursos humanos puesto que el proyecto presentado requiere un número de horas de trabajo elevadas para realizar el diseño del mismo, así como implementarlo en un entorno de desarrollo para confirmar su validez. El resto de costes presentados se corresponden con elementos básicos que son requeridos en la mayoría de proyectos de ingeniería (Ordenadores, Luz, Internet...etc.). A modo de conocer el impacto de cada apartado sobre los costes, se incluye el siguiente gráfico:

FIGURA 9.1: Resumen de costes del proyecto

10 Conclusiones

La principal conclusión del trabajo desarrollado es que se ha conseguido alcanzar los objetivos planteados para el mismo, cumpliendo con los plazos previstos, y con el coste presupuestado inicialmente. Mediante este trabajo se ha definido el diseño de un modelo que permita la comercialización de licencias de forma segura, haciendo para ello uso de diferentes tecnologías y esquemas de seguridad. Los mecanismos presentados en el diseño son ampliamente utilizados en el mercado, lo que permite asegurar un mejor soporte de la plataforma. Por otro lado, la flexibilidad aportada por esta solución permite que los diferentes proveedores, distribuidores y/o gestores de pago puedan realizar la implementación del sistema con total libertad, introduciendo las funcionalidades extra que se requieran dentro de su arquitectura.

A modo de realizar una primera validación del modelo y comprobar que las características y objetivos definidos para el mismo se cumplen, se ha simulado un escenario dentro de un entorno de desarrollo. La programación de la lógica de aplicación de los diferentes elementos que componen el modelo se ha realizado en base al uso de tecnologías que permitan un despliegue rápido y que cumplan con las especificaciones propuestas.

El proyecto actualmente está centrado en la venta y gestión *online* de licencias *software*. Sin embargo, existen otros sectores de mercado dentro del comercio electrónico donde se podrían implementar mecanismos de seguridad similares que permitiesen dar un mayor de confianza a los usuarios, generando un volumen de ventas mayor y beneficiando así a los comerciantes.

Bibliografía

[1] V. Rosas. (2017). Cuestionan a representante de G2A por acciones fraudulentas, dirección: http://www.levelup.com/noticias/418964/Cuestionan-a-representante-de-G2A-por-acciones-fraudulentas.

[2] T. Baccam, "Making Database Security an IT Security Priority", 2009. dirección: https://www.sans.org/reading-room/whitepapers/analyst/making-database-security-security-priority-34835.

[3] P. Rubens. (2016). 7 Database Security Best Practices, dirección: https://www.esecurityplanet.com/network-security/6-database-security-best-practices.html.

[4] J. Ellingwood. (2015). 7 Security Measures to Protect Your Servers, dirección: https://www.digitalocean.com/community/tutorials/7-security-measures-to-protect-your-servers.

[5] D. Winner, "Making Your Network Safe for Databases", 2002. dirección: https://www.sans.org/reading-room/whitepapers/application/making-network-safe-databases-2.

[6] J. Wang y Z. A. Kissel, *Introduction to Network Security: Theory and Practice*. John Wiley & Sons, 2015. dirección: https://www.amazon.com/Introduction-Network-Security-Theory-Practice/dp/1118939484.

[7] H. Patel. (2016). What is the difference between packet firewall, stateful firewall and application firewall?, dirección: https://www.quora.com/What-is-the-difference-between-packet-firewall-stateful-firewall-and-application-firewall.

[8] K. Nasser. (2013). How To Setup a Firewall with UFW on an Ubuntu and Debian Cloud Server, dirección: https://www.digitalocean.com/community/tutorials/how-to-setup-a-firewall-with-ufw-on-an-ubuntu-and-debian-cloud-server.

[9] A. Umar, *Information Security and Auditing in the Digital Age*. NGE Solutions, 2003. dirección: https://www.amazon.com/Information-Security-Auditing-Digital-Age/dp/097274147X/.

[10] J. Martin. (2016). Security Nuggets : Encrypting SQL Server Connections, dirección: https://blogs.sentryone.com/johnmartin/security-nuggets-encrypting-connections/.

[11] S. Biswas. (2018). A Guide to SSH Port Forwarding/Tunnelling, dirección: https://www.booleanworld.com/guide-ssh-port-forwarding-tunnelling/.

[12] E. Davis. (2016). Understanding 4 Database Types, for Ecommerce, dirección: https://www.practicalecommerce.com/Understanding-4-Database-Types-for-Ecommerce.

[13] C. Roe. (2012). ACID vs. BASE: The Shifting pH of Database Transaction Processing, dirección: http://www.dataversity.net/acid-vs-base-the-shifting-ph-of-database-transaction-processing/.

[14] M. Winand, *SQL Performance Explained: Everything Developers Need to Know about SQL Performance*. M. Winand, 2012. dirección: https://www.amazon.com/gp/product/3950307826/ref=as_li_ss_tl?ie=UTF8&camp=1789&creative=390957&creativeASIN=3950307826&linkCode=as2&tag=winandat-20&m=A1PG4MQR22GK8T.

[15] O. Blancarte. (2017). Escalabilidad Horizontal y Vertical, dirección: https://www.oscarblancarteblog.com/2017/03/07/escalabilidad-horizontal-y-vertical/.

[16] J. Serra. (2015). Relational databases vs Non-relational databases, dirección: http://www.jamesserra.com/archive/2015/08/relational-databases-vs-non-relational-databases/.

[17] C. Birgen, "SQL vs. NoSQL", Norwegian University of Science y Technology, inf. téc., 2014. dirección: http://folk.ntnu.no/preisig/HAP_Specials/AdvancedSimulation_files/2014/AdvSim-2014__Birgen_Cansu_Databases.pdf.

[18] R. Fernandez. (2014). Bases de datos NoSQL. Elige la opción que mejor se adapte a tus necesidades, dirección: https://www.genbetadev.com/bases-de-datos/bases-de-datos-nosql-elige-la-opcion-que-mejor-se-adapte-a-tus-necesidades.

[19] M. Chapple. (2018). Abandoning ACID in Favor of BASE in Database Engineering, dirección: https://www.lifewire.com/abandoning-acid-in-favor-of-base-1019674.

[20] A. Mehra. (2017). Understanding the CAP Theorem, dirección: https://dzone.com/articles/understanding-the-cap-theorem.

[21] R. Fernandez. (2014). NoSQL: clasificación de las bases de datos según el teorema CAP, dirección: https://www.genbetadev.com/bases-de-datos/nosql-clasificacion-de-las-bases-de-datos-segun-el-teorema-cap.

[22] S. Martín. (2017). Bases de datos NoSQL: La guía definitiva, dirección: https://blog.pandorafms.org/es/bases-de-datos-nosql/.

[23] A. Kamaraju. (2011). Native DB Encryption Versus Third-Party Enterprise Encryption - What's the Difference?, dirección: http://www.dbta.com/Editorial/Think-About-It/Native-DB-Encryption-Versus-Third-Party-Enterprise-Encryption---Whats-the-Difference-78833.aspx.

[24] H. C. van Tilborg, *Encyclopedia of Cryptography and Security*. Springer, 2011. dirección: https://www.springer.com/gp/book/9781441959058?utm_campaign=3_pier05_buy_print&utm_content=en_08082017&utm_medium=referral&utm_source=google_books#aboutAuthors.

[25] P. M. K. Tepper. (2012). Desmitificando la Encriptación (Parte I), dirección: https://msdn.microsoft.com/es-es/library/bb972216.aspx.

[26] A. Mehmood y C. Allen. (2017). Key Management and use cases for HSMs, dirección: https://www.cryptomathic.com/news-events/blog/key-management-and-use-cases-for-hsms.

[27] G. McCracken. (2016). Primary Distinguishers between HSM and Key Manager, dirección: https://www.winmagic.com/blog/2016/04/28/primary-distinguishers-hsm-key-manager/.

[28] Bobby. (2017). How Does HTTPS Work? SSL/TLS Explained, dirección: https://tiptopsecurity.com/how-does-https-work-ssl-tls-explained/.

[29] R. Harifa. (2017). Difference Between TLS and SSL, dirección: http://www.differencebetween.net/technology/difference-between-tls-and-ssl/#ixzz5H0Qr3nyK.

[30] Auth0. (2015). Introduction to JSON Web Tokens, dirección: https://jwt.io/introduction/.

[31] T. Tkalec. (2014). JSON Web Token Tutorial: An Example in Laravel and AngularJS, dirección: https://www.toptal.com/web/cookie-free-authentication-with-json-web-tokens-an-example-in-laravel-and-angularjs.

[32] Auth0. (2016). Where to Store Your JWTs, dirección: https://auth0.com/docs/security/store-tokens#where-to-store-your-jwts.

[33] P. Otemuyiwa. (2016). JSON Web Tokens vs. Session Cookies: In Practice, dirección: https://ponyfoo.com/articles/json-web-tokens-vs-session-cookies.

[34] ExpressJS. (2014). Express - Session: Session Store Implementation, dirección: https://www.npmjs.com/package/express-session.

[35] J. Ciolek. (2016). JSON Web Tokens (JWT) vs Sessions, dirección: https://float-middle.com/json-web-tokens-jwt-vs-sessions/.

[36] Órgano de Jefatura del Estado. (2018). Ley Orgánica de Protección de datos de Carácter Personal, dirección: https://ayudaleyprotecciondatos.es/2017/06/27/lopd-2018/#Articulo_1_Objeto_de_la_ley.

[37] ——, (2014). Ley de servicios de la sociedad de la información y de comercio electrónico, dirección: https://www.boe.es/buscar/act.php?id=BOE-A-2002-13758.

[38] ——, (2018). Ley de Propiedad Intelectual, dirección: https://www.boe.es/buscar/act.php?id=BOE-A-1996-8930.

Anexo I

I - Pliego de condiciones

El presente pliego de condiciones tiene por objetivo determinar los requisitos mínimos aceptables para el diseño y la ejecución del proyecto presentado en este documento, así como definir las características, trabajos y elementos *hardware* y/o *software* necesarios para ello.

Ámbito de aplicación

El pliego de condiciones se centra en los requisitos del proyecto para el diseño de un modelo de comercialización de licencias *software* seguro. Por tanto, todos los puntos relacionados con el mismo serán enfocados según las características y necesidades de las empresas dedicadas a este tipo de actividades comerciales.

Normativa de aplicación

Además de las características técnicas descritas en este documento, el hecho de trabajar con información sensible referente a los usuarios finales y a las licencias obliga a tener en cuenta y aplicar las siguientes normativas relacionadas con la protección de datos y la comercialización de activos en la red (para más detalles sobre la normativa vigente, consultar **II - Normativa aplicable**):

- **Ley Orgánica de Protección de datos de Carácter Personal:** El hecho de requerir datos sensibles de los usuarios para poder desarrollar la actividad comercial supone que la protección y uso de los mismos debe regirse por la ley citada en este punto.
- **Ley de Servicios de la Sociedad de la Información y del Comercio Electrónico:** El tipo de organizaciones a las que va dirigido este proyecto son aquellas que se dedican principalmente a la compraventa de licencias *software* a través de la red. Por tanto, las actividades llevadas a cabo por estas empresas deben estar reguladas por medio de la ley de comercio electrónico.
- **Ley de la Propiedad Intelectual:** Las licencias comercializadas permiten el acceso a contenido protegido con derechos de autor. Por lo tanto, estos activos deben respetar el conjunto de derechos que corresponden a los autores por sus obras, y deben comercializarse según rige esta ley.

Condiciones en los equipos *hardware*

El desarrollo del proyecto conlleva realizar la virtualización de diferentes equipos para poder construir una maqueta que verifique la validez del modelo teórico presentado con anterioridad. Por tanto, los equipos utilizados para la programación e implementación del sistema deben ser lo suficientemente potentes para poder soportar la emulación de los tres servicios que constituyen la maqueta[1] con un rendimiento óptimo. Estas características están centradas principalmente en una capacidad de cálculo elevada, así como una cantidad de memoria RAM alta.

Condiciones en las herramientas *software*

El proyecto desarrollado, aún disponiendo de un gran peso a nivel de contenido teórico, requiere la implementación práctica de los diferentes elementos que se han descrito a lo largo del documento con el fin de determinar si el diseño realizado es correcto. Por tanto, a la hora de considerar la programación de la maqueta, se deben tener en cuenta los siguientes apartados a nivel *software*:

- El entorno de desarrollo será constituido por un sistema Linux, concretamente **Fedora 26**, donde se instalarán todos los elementos *software* requeridos para el desarrollo y el posterior despliegue.
- Se precisa de un editor de código que facilite la programación de los diferentes módulos. Concretamente se hará uso de la herramienta **Visual Studio Code** proporcionada por Microsoft puesto que es compatible con diversos lenguajes de programación y dispone de diferentes *plugins* que permiten gestionar el código de manera más sencilla.
- Finalmente, resulta necesaria la instalación de los elementos que van a constituir el *backend* de cada elemento del modelo y que van a dar respuesta a los servicios demandados. En este caso, es necesario disponer de la plataforma **NodeJS** para albergar la lógica de las aplicaciones web, así como el motor de bases de datos **MongoDB** para almacenar todos los datos generados por los servidores web.

Condiciones generales

Las condiciones que se presentan a continuación están diseñadas para obtener un desempeño eficaz en el modelo, así como garantizar en todo momento la seguridad en las operaciones realizadas dentro de los diferentes elementos del sistema:

- Los datos de usuario deben estar siempre correctamente protegidos atendiendo a la legislación vigente sobre protección de datos. Por otro lado, las licencias deben ser igualmente protegidas, puesto que estos activos dan acceso a contenidos protegidos por derechos de autor. El proveedor es el responsable de las mismas, y otorgará al distribuidor unicamente la información necesaria para su comercialización a clientes finales.

[1] Proveedor de licencias, distribuidor y gestor de pagos

- Además del almacenamiento, las comunicaciones a realizar entre las diferentes partes del modelo deben estar correctamente securizadas para evitar que la información transmitida se vea comprometida.
- El gestor de pagos actúa como un elemento fundamental en el sistema. Este agente vinculará los pagos realizados a las licencias, de forma que en ningún momento se pueda cometer un fraude a la hora de comercializarlas.

Pruebas y verificaciones

Para finalizar, se van a ejecutar una serie de pruebas sobre el sistema desarrollado para validar que las diferentes características y objetivos definidos para el proyecto se cumplan satisfactoriamente (véase **III - Plan de pruebas**).

II - Normativa aplicable

Tal y como se ha indicado en el pliego de condiciones, el desarrollo del siguiente proyecto debe considerar tres normativas principalmente. A continuación se resumen los puntos más importantes referentes a estas leyes:

1. Ley Orgánica de Protección de datos de Carácter Personal (LOPD)

Objeto de la ley

La Ley Orgánica de Protección de datos de Carácter Personal 15/1999 de 13 de diciembre (LOPD), tiene por objeto proteger y garantizar las libertades y los derechos fundamentales de las personas físicas, su honor e intimidad personal y familiar [36].

La LOPD establece unas obligaciones en relación a la protección de datos de carácter personal contenidos en ficheros automatizados y no automatizados (en papel) que poseen empresas y administraciones públicas, y que son tratados por éstas con diferentes finalidades; gestión personal, proveedores, clientes, campañas de marketing..etc. Esta ley ha sido recientemente actualizada según el ordenamiento jurídico del Reglamento (UE) 2016/679 del Parlamento Europeo y el Consejo, de 27 de abril de 2016, constituyéndose así el Reglamento General de Protección de Datos (RGPD).

Obligaciones básicas de las empresas

- **Declarar los ficheros** automatizados y no automatizados con datos de carácter personal a la Agencia de Protección de Datos.
- **Legitimar los datos** personales que se disponen mediante el cumplimiento de los siguientes principios:
  - **Principio del consentimiento del afectado**
  - **Principio de información al afectado**
  - **Principio de calidad de los datos**
- **Proteger los ficheros** automatizados y no automatizados para preservar la confidencialidad, integridad y disponibilidad de los datos siguiendo lo establecido en el Reglamento de Seguridad.
- Como aspectos básicos se requiere:
  - **Disponer de un documento de seguridad**
  - **Definir e implantar los Procedimientos requeridos**
  - **Nombre a un responsable de seguridad**
  - **Formar y concienciar en materia de seguridad de la información a todo el personal que gestione datos personales**

- **Cumplir con las Medidas de Seguridad** según el tipo o nivel de datos que disponga. Determinadas en el Real Decreto 1720/2007, de 21 de diciembre, por el que se aprueba el Reglamento de desarrollo de la Ley Orgánica 15/99, de 13 de Diciembre, de Protección de Datos de Carácter Personal.

Tipos y niveles de datos

Los datos de carácter personal se clasifican en tres niveles atendiendo a su naturaleza y finalidad. A cada nivel le corresponden unas medidas de seguridad, siendo las de nivel alto las más estrictas:

- **Nivel básico:** Todos los datos de carácter personal estarán, como mínimo, en este nivel. A todos los ficheros con datos personales se les deberán aplicar las medidas de seguridad de nivel básico.
- **Nivel medio:** En general, los datos relativos a infracciones administrativas o penales, sobre solvencia patrimonial o crédito sobre servicios financieros, y aquellos que permitan determinar un perfil de las personas. A estos datos será de aplicación las medidas de nivel básico y las de nivel medio.
- **Nivel alto:** En general, los datos relativos a ideología, afiliación sindical, religión, creencias, origen racial, salud o vida sexual. A estos datos se les aplicarán las medidas definidas para el nivel básico, el medio y el alto.

Medidas de seguridad obligatorias

- **Medidas de seguridad para el Nivel Básico:** Se deben tener presentes las siguientes medidas de seguridad:
  - **Documento de seguridad**
  - **Definición de funciones del personal**
  - **Registro de incidencias**
  - **Identificación y autenticación de usuarios (Control de acceso)**
  - **Gestión y soporte de documentos**
  - **Mantenimiento de copias de seguridad y soportes informáticos**
- **Medidas de seguridad para el Nivel Medio:** Incluye las medidas indicadas en el nivel anterior, añadiendo ademas un responsable de seguridad, auditoría bienal y control de acceso físico.
- **Medidas de seguridad de nivel Alto:** Incluye los puntos indicados en los niveles anteriores, más las siguientes características:
  - **Copias externalizadas**
  - **Registros de acceso (Almacenarlos al menos durante 24 meses)**
  - **Cifrado de copias**
  - **Cifrado de las comunicaciones**

2. Ley de Servicios de la Sociedad de la Información y del Comercio Electrónico

La presente Ley tiene como objeto la incorporación al ordenamiento jurídico español de la Directiva 2000/31/CE, del Parlamento Europeo y del Consejo, de 8 de junio, relativa a determinados aspectos de los servicios de la sociedad de la información, en particular, el comercio electrónico en el mercado interior (Directiva sobre el comercio electrónico) [37].

Aviso Legal

Las empresas de comercio electrónico deben mostrar, en su sitio web, determinada información sobre ellas mismas que debe permanecer allí de forma permanente y fácilmente accesible. Entre la información que la Ley de Comercio Electrónico obliga a las empresas a hacer pública se encuentran:

- **Su nombre**
- **El domicilio social de la empresa**
- **La dirección de correo electrónico**
- **Su Número de Identificación Fiscal (NIF)**
- **Los datos de inscripción en el Registro Mercantil**

Obligaciones de las empresas

La empresa que ofrece sus productos o servicios a los clientes en la red está obligada, según la Ley de Comercio Electrónico, a informar a los usuarios a cerca de todos los trámites que se seguirán en el proceso de compra o contratación. Del mismo modo, el proveedor está obligado a informar acerca del archivo del documento electrónico resultante del proceso de compra o contratación. Pero las obligaciones de la empresa no finalizan cuando lo hace la transacción comercial. La Ley de Comercio Electrónico estipula que es necesario informar al cliente, mediante un acuse de recibo, que la transacción se ha realizado con éxito.

Derechos de los consumidores

La norma fija desde el derecho de los usuarios a que la información que reciben acerca de un producto o servicio sea veraz y completa, hasta el derecho a recibir un presupuesto previo a la transacción. Otros de los derechos que se estipulan alrededor del comercio electrónico son:

- **El derecho a recibir una copia del contrato o una factura de la compra**
- **El derecho a obtener una garantía sobre los bienes comprados**
- **El derecho a disponer de un servicio técnico**
- **El derecho a que los productos superen los controles de calidad y seguridad que la legislación española y europea contemplan**

Protección de datos

Otro de los derechos básicos de cualquier usuario o consumidor es que se respeten y protejan sus datos personales. Por ello, y para ello, cualquier persona deberá dar su consentimiento para la recolección de estos datos personales. Pero además, según la Ley de Comercio Electrónico, para expresar su autorización se deberá informar previamente al usuario de por qué, para qué, cómo y por quién van a ser recogidos, tratados, filtrados y utilizados sus datos personales, especialmente si esa información pasará, posteriormente, a manos de terceros.

Envío de comunicaciones comerciales

Tampoco se podrá, sin consentimiento expreso del usuario, enviar comunicaciones comerciales, ya sean en forma de publicidad, promociones o cualquier otro tipo de mensaje. Estas comunicaciones, según la Ley de Comercio Exterior, además de ser plenamente identificables como comerciales, sin posibilidad de que pueda inducir a error, deben proteger, también, los datos de sus destinatarios.

3. Ley de la Propiedad Intelectual

La disposición final segunda de la Ley 27/1995, de 11 de octubre, de incorporación al Derecho español de la Directiva 93/98/CEE, del Consejo, de 29 de octubre, relativa a la armonización del plazo de protección del derecho de autor y de determinados derechos afines, autorizó al Gobierno para que, antes del 30 de junio de 1996, aprobara un texto que refundiese las disposiciones legales vigentes en materia de propiedad intelectual, regularizando, aclarando y armonizando los textos que hubieran de ser refundidos [38].

Definiciones

La propiedad intelectual de una obra literaria, artística o científica corresponde al autor por el sólo hecho de crearla. Se considera autor a la persona natural que crea alguna obra artística, científica o literaria. Se presumirá autor, salvo prueba de lo contrario, a quien aparezca como tal en la obra, mediante su nombre, firma o signo que lo identifique. La divulgación de una obra es toda expresión de la misma que, con el consentimiento del autor, la haga accesible por primera vez al público en cualquier forma. El objeto de la propiedad intelectual son las creaciones originales literarias, artísticas o científicas expresadas por cualquier medio o soporte, tangible o intangible, actualmente conocido o que se invente en el futuro.

Derechos y Deberes

La propiedad intelectual está integrada por derechos de carácter personal y patrimoniales, que atribuyen al autor la plena disposición y el derecho exclusivo de los derechos de explotación de su obra en cualquier forma y, en especial, los derechos de reproducción, distribución, comunicación pública y transformación, que no podrán ser realizadas sin su autorización, salvo en los casos previstos en la Ley. El derecho a la propiedad intelectual tiene unos límites. Las obras ya divulgadas podrán reproducirse sin autorización del autor en los siguientes casos:

- **Como consecuencia o para constancia en un procedimiento judicial o administrativo**
- **Para uso privado del copista y siempre que la copia no sea objeto de utilización colectiva ni lucrativa**
- **Para uso privado de invidentes**

Es lícita la inclusión en una obra propia de fragmentos de obras ajenas de naturaleza escrita, sonora o audiovisual, así como la de obras aisladas de carácter plástico, fotográfico, figurativo o análogo, siempre que se trate de obras ya divulgadas y su inclusión se realice a título de cita o análisis, comentario o juicio crítico. Tal utilización sólo podrá realizarse con fines docentes o de investigación e indicando la fuente y el nombre del autor de la obra utilizada. Sólo con el exclusivo fin de informar sobre la actualidad, se podrán reproducir, distribuir y comunicar las conferencias, alocuciones, informes ante los Tribunales y otras obras del mismo carácter que se hayan pronunciado en público.

La extinción de los derechos de explotación de las obras determinará su paso al dominio público. Las obras de dominio público podrán ser utilizadas por cualquiera, siempre que se respete la autoría y la integridad de la obra.

Características

El derecho de la propiedad intelectual tiene las siguientes características:

- **Los derechos de explotación de la obra se transmiten, «mortis causa», por cualquiera de los medios admitidos en derecho**
- **También pueden transmitirse por actos «inter vivos». La Ley de Propiedad Intelectual legitima al titular de los derechos por ella reconocidos para instar el cese de la actividad ilícita del infractor y exigir la indemnización de los perjuicios materiales y morales causados**
- **Podrán ser objeto de inscripción en el Registro los derechos de propiedad intelectual relativos a las obras y demás producciones protegidas por la Ley de Propiedad Intelectual**
- **El titular o cesionario en exclusiva de un derecho de explotación sobre una obra o producción protegidas por la Ley podrá anteponer a su nombre el símbolo, © con precisión del lugar y año de la divulgación de aquellas.**

III - Plan de pruebas

Todo desarrollo *software* requiere un apartado dedicado al diseño y realización de diferentes pruebas para constatar que la programación de los módulos se ha realizado correctamente y que las especificaciones y objetivos propuestos se cumplen.

El desarrollo de la maqueta no ha sido definida para generar un sistema completo que este preparado para lanzarse a un entorno de producción, sino que se ha centrado en incluir los elementos básicos necesarios para probar los procedimientos y mecanismos que se han definido en el apartado de **Descripción de la solución**. En primer lugar se definirán las características del entorno de pruebas para conocer las especificaciones *hardware/software* donde se ha desplegado el modelo y posteriormente, se describirán las pruebas llevadas a cabo en el sistema:

Especificaciones del entorno de pruebas

El desarrollo y despliegue de los diferentes elementos que conforman el modelo se ha realizado dentro de un ordenador portátil que dispone de las siguientes especificaciones *hardware*:

Modelo	ASUS ZenBook UX330UA
Procesador	Intel Core I7 7500U 2.7 GHz (soporta aceleración VT-x para maquinas virtuales)
Memoria RAM	8 GB LPDDR3 1866MHz SDRAM
Almacenamiento	256GB SATA3 M.2 SSD
Conectividad	WiFi 802.11AC (hasta 867 Mbps)

CUADRO I.1: Especificaciones *hardware* del entorno de pruebas

Tal y como se puede observar, el equipo utilizado dispone de unas especificaciones adecuadas para desplegar los diferentes servicios dentro del mismo. En lo referente al apartado *software*, los equipos que conforman cada uno de los agentes en el modelo son virtualizados dentro de un entorno de trabajo Linux, concretamente Fedora 26. Dentro de esta distribución se han instalado igualmente las herramientas necesarias para hacer frente al despliegue de los servicios (MongoDB, NodeJS, Visual Studio Code, Postman...etc.).

El hecho de utilizar una maquina virtual en vez de desplegar los servicios de forma nativa supone que va a haber una reducción en el rendimiento ofrecido. Sin embargo, el equipamiento utilizado es suficientemente potente como para hacer frente a la virtualización y este escenario va a permitir conocer el rendimiento proporcionado dentro de un entorno más desfavorable[2]. A continuación se presentan las diferentes pruebas que se han realizado para verificar la validez del modelo y medir de manera aproximada el rendimiento proporcionado por el sistema:

[2]Hay que tener presente igualmente que el hecho de trabajar en un ámbito local implica que no hay retardos de transmisión. Dentro de un entorno real hay que considerarlos, aunque son independientes al proyecto realizado

Prueba I - Comprobación de las licencias previo pago

Este punto es uno de los más importantes dentro del modelo propuesto, puesto que esta característica va a permitir al usuario final poder comprobar el estado de la licencia sin la necesidad de haber pagado por ella previamente. Para verificar el correcto funcionamiento de esta funcionalidad, así como comprobar que el rendimiento es óptimo, se presentan diferentes casos de prueba con la introducción de varias licencias. El procedimiento realizado se describe en la tabla siguiente:

Condiciones a cumplir	El usuario accede a la sección de compra del distribuidor y este le proporciona la licencia a verificar. Cada vez que acceda, se le entregará una licencia distinta y se probará el procedimiento en todos los casos
Casos de prueba	1. Introducción de un código aleatorio que no se corresponda con una licencia 2. Introducción de una licencia que no se ha consultado ni registrado con anterioridad 3. Introducción de la misma licencia 4. Introducción de una licencia previamente registrada
Herramientas necesarias	Servidores web virtualizados y navegador web
Resultados esperados	En función del estado de la licencia el proveedor debe mostrar una ventana informativa al usuario indicando que debe hacer
Resultados obtenidos	Los resultados son los esperados y se corresponden con el funcionamiento correcto del sistema. La transacción se realiza de manera casi instantánea (70ms), por lo que la interacción entre el *backend* y el *frontend* es adecuada

CUADRO I.2: Prueba I: Comprobación de las licencias previo pago

En la siguiente tabla se muestran los mensajes generados para cada uno de los casos de prueba que se han definido anteriormente:

Caso 1	La licencia que ha introducido no se encuentra registrada dentro de la base de datos. Compruebe que ha escrito correctamente el código o solicite una nueva licencia al distribuidor
Caso 2	La licencia es correcta. Puede continuar con su proceso de compra
Caso 3	La licencia ha sido previamente consultada en la fecha <XXXX>. Si no ha sido usted, puede ocurrir que otro usuario haya decidido cancelar el proceso de compra, o si la ha comprado, todavía no la ha registrado. Si continua con el proceso de compra, hágalo bajo su responsabilidad
Caso 4	La licencia ha sido previamente registrada por otro usuario. Si no ha sido usted, solicite al distribuidor una nueva licencia y no continué con el proceso de compra

CUADRO I.3: Resultados obtenidos para la prueba I

Prueba II - Protección de rutas

Algunas de las funcionalidades propuestas en este modelo deben estar protegidas para que solo los usuarios que se encuentren autenticados puedan acceder a las mismas. Concretamente, dentro del proveedor se encuentran protegidas las rutas para la verificación y registro de licencias, dentro del distribuidor las funciones de compra, y el gestor de pagos se encuentra totalmente cerrado para que solo pueda ser utilizado por empresas autorizadas. Los procedimientos seguidos en esta prueba se representan en la siguiente tabla:

Condiciones a cumplir	El usuario final o agente externo solo puede acceder a las funcionalidades mencionadas si envía en las cabeceras la información correspondiente a la sesión
Casos de prueba	1. Acceso a las funciones de verificación y registro en el proveedor sin iniciar sesión 2. Acceso a la sección de compra en el distribuidor sin iniciar sesión 3. Consulta al gestor de pagos sin proporcionar identidad
Herramientas necesarias	Servidores web virtualizados, navegador web, *software* Postman
Resultados esperados	Para todos los casos propuestos se debe denegar el acceso a los servicios
Resultados obtenidos	Los resultados son los esperados, puesto que no se pueden acceder a las funciones mencionadas

CUADRO I.4: Prueba II: Protección de rutas

Por medio del *software* Postman, se han generado diferentes solicitudes a los servicios que se han mencionado en cada punto de tabla anterior. Los resultados obtenidos como respuesta a las solicitudes se representan en la siguiente tabla:

Caso 1	Se produce una redirección y se carga la página web principal del proveedor para iniciar sesión
Caso 2	Se produce una redirección y se carga la página web principal del distribuidor para iniciar sesión
Caso 3	Se obtiene un código de error 401 *Unauthorized*

CUADRO I.5: Resultados obtenidos para la prueba II

Prueba III - Verificación de *tickets*

Este punto resulta necesario para que el proveedor pueda verificar que el usuario final ha realizado la compra de la licencia y se ha generado un *ticket* vinculado a la misma. Cuando el usuario finalice la compra al distribuidor, dentro del gestor de pagos se generará un *ticket*, el cual será entregado al cliente para certificar el pago de la licencia. Con dicho *ticket* y el código de la licencia encriptada, se realizará una consulta al gestor de pago que comprobará que la información enviada es correcta. Tal y como se ve en la siguiente tabla, se presentan diferentes casos de prueba:

Condiciones a cumplir	El gestor de pagos indicará mediante un código de estado al proveedor si los datos introducidos son correctos. Si es así, la licencia será desencriptada y entregada al usuario final
Casos de prueba	1. Introducción de un código de licencia falso 2. Introducción de una licencia válida y un *ticket* falso 3. Introducción de un código de licencia válido y un *ticket* referente al pago de otra licencia[3] 4. Introducción de una licencia correcta junto con un *ticket* vinculado a la misma
Herramientas necesarias	Servidores web virtualizados, navegador web
Resultados esperados	Unicamente en caso de introducir datos válidos se debe entregar la licencia desencriptada al usuario
Resultados obtenidos	Los resultados obtenidos se corresponden con el funcionamiento esperado en el sistema

CUADRO I.6: Prueba III: Verificación de *tickets*

En función de los diferentes casos propuestos, se obtienen las siguientes respuestas por parte del proveedor:

Caso 1	El proveedor no realiza la consulta al gestor puesto que detecta que la licencia introducida no es válida, indicándole al usuario que los datos indicados no son correctos
Caso 2	El gestor de pagos indica al proveedor que no existe el *ticket* introducido, mostrando al usuario un mensaje de error solicitando que incluya información válida referente al pago
Caso 3	El gestor de pagos indica al proveedor que el *ticket* se encuentra vinculado al pago de una licencia distinta. El proveedor muestra, por tanto, un mensaje al cliente indicando que proporcione el *ticket* asociado al pago de dicha licencia
Caso 4	El gestor de pago informa de la validez de los datos proveedor y este muestra al usuario la licencia final desencriptada

CUADRO I.7: Resultados obtenidos para la prueba III

[3]Puede darse el caso de que un usuario aproveche la compra de una licencia para utilizar el *ticket* generado para validar otras licencias. La forma de evitar este fraude es vincular cada licencia a un *ticket* único.

Prueba IV - Comportamiento presentado ante varias consultas simultaneas

El hecho de que cada usuario vaya a recibir una licencia encriptada en el momento de acceder a la sección de compra hace que sea importante controlar la atomicidad de las operaciones para evitar que a varios usuarios se les muestre el mismo código y que esto de lugar a problemas en el registro de las licencias. En esta prueba, se va a hacer uso del *software* Postman Runner para generar diferentes consultas simultaneas y comprobar que la licencia entregada en cada una de ellas sea distinta. La tabla mostrada a continuación resume los requisitos de la prueba:

Condiciones a cumplir	El distribuidor debe gestionar las peticiones de forma que la operación para obtener una licencia de la base de datos se realice de forma completa, entregando así una licencia distinta para cada petición
Casos de prueba	Generar 3 consultas simultaneas hacia la ruta disponible para la compra de licencias, considerando que dentro del distribuidor unicamente hay 2 licencias disponibles
Herramientas necesarias	Servidores web virtualizados, *software* Postman Runner
Resultados esperados	Para cada consulta realizada se debe devolver una licencia encriptada distinta
Resultados obtenidos	El resultado obtenido es el esperado, puesto que cada consulta dispone de una licencia distinta

CUADRO I.8: Prueba IV: Comportamiento presentado ante varias consultas simultaneas

En cuanto a los resultados obtenidos, en la primera y segunda consulta se obtienen las dos licencias respectivamente, una para cada consulta, mientras que para la tercera se indica que no existe *stock* suficiente. Este comportamiento es el esperado dentro de la situación planteada por lo que se puede asegurar que el sistema es coherente con las peticiones.

Prueba V - Rendimiento general del sistema

Para acabar con el plan de pruebas, resulta interesante realizar un análisis sobre el rendimiento que aporta el sistema en cuanto a los tiempos de respuesta. La maqueta que ha sido elaborada en este proyecto para verificar la validez del modelo ha sido implementada dentro de un entorno Linux virtualizado, lo que implica que el rendimiento obtenido va a ser menor que si esta fuera implementada dentro de equipamiento nativo.

Para comprobar los tiempos de respuesta se va a hacer uso una vez más del *software* Postman Runner. El *test* debe ser realizado en aquellas partes que suponen una carga de trabajo más elevada para los servidores. En este caso, se ejecutarán las pruebas en los siguientes puntos:

- **Registro de licencias:** Esta operación requiere tanto la verificación de los *tickets* por parte del gestor de pago como la desencriptación de la licencia.
- **Generación de *ticket* de pago:** Cuando se realiza la compra de una licencia, el gestor de pago debe registrar dicho pago y generar un *ticket*.

Mediante la realización de diferentes consultas iterativas, se obtienen los siguientes tiempos de respuesta para cada operación:

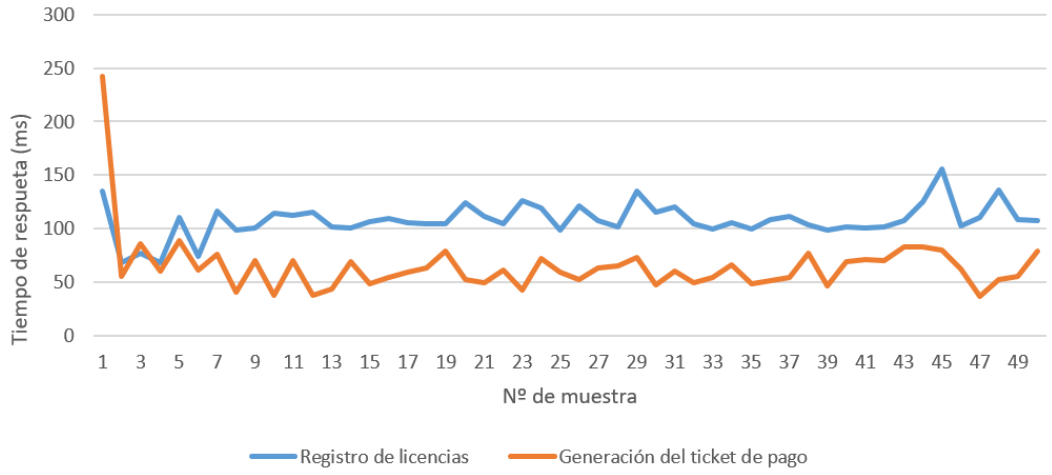


FIGURA I.1: Tiempos de respuesta para cada operación

Como es lógico, la operación de verificación y registro supone un tiempo mayor puesto que deben realizarse más acciones. Teniendo en cuenta que se está trabajando en un entorno virtualizado con varios servidores corriendo simultáneamente, el tiempo medio de respuesta obtenido resulta adecuado y la experiencia de usuario ofrecida es óptima. Sin embargo, nuevamente el hecho de estar trabajando en un entorno local implica que no se tienen en cuenta los retardos que puedan producirse por la transferencia de las peticiones y respuestas a través de Internet.

Anexo II

Diseño de bajo nivel: Maqueta para la validación del modelo

Tal y como se ha venido anunciando a lo largo del documento, se ha implementado una maqueta dentro de un entorno de desarrollo que permita verificar las funcionalidades que se han descrito a lo largo del apartado de **Descripción de la solución**. Las características contempladas dentro del despliegue realizado distan del conjunto de funcionalidades disponibles en una plataforma de comercio *online*, pero resultan suficientes para comprobar la validez del modelo teórico definido. Los tres elementos que conforman la maqueta han sido virtualizados dentro de un mismo equipo en un entorno de trabajo Linux, para permitir un despliegue mucho más rápido[1].

Puesto que el almacenamiento de los datos se va a realizar por medio de una base de datos no relacional (**MongoDB**), se ha decidido utilizar el *stack* de desarrollo **MEAN** para realizar la programación de la lógica de aplicación que incluye cada uno de los elementos que conforman el modelo[2]. Esta solución aprovecha el uso del lenguaje *JavaScript* para implementar las funcionalidades correspondientes tanto a nivel de *frontend* como de *backend*. Puesto que el modelo esta constituido por varias partes (proveedor de licencias, distribuidor y gestor de pagos), se va a describir la lógica de cada elemento en cada uno de los siguientes subapartados:

Proveedor de licencias

El proveedor de licencias dispone de diferentes funcionalidades, ya sea para la administración del contenido de la base de datos como para ofrecer los servicios de verificación y registro final de licencias. Se explican a continuación cada uno de los puntos existentes en la lógica de aplicación:

[1] El diseño de la lógica de aplicación, así como el almacenamiento de datos es idéntico al presentado en el modelo teórico. Sin embargo, en lo referente a las técnicas de seguridad, se han implementado unicamente los mecanismos básicos para asegurar las transacciones (HTTPS, control de sesiones y protección de rutas, cifrado a nivel de aplicación)

[2] En este anexo se incluye el desarrollo de aquellos métodos más representativos dentro del modelo.

Administrador de licencias

La característica más relevante en este punto es poder ofrecer un sistema para la solicitud de datos de sesión a la hora de acceder a dicha ruta. Para ello, se ha implementado el siguiente código:

```
var auth = req.headers['authorization']; //RECOGER DATOS DE AUTENTICACIÓN DE LAS CABECERAS
if(!auth) { //NO HAY DATOS SOBRE AUTENTICACIÓN
 res.statusCode = 401;
 res.setHeader('WWW-Authenticate', 'Basic realm="BD administration page. Only for developers"');
 res.end('<html><body>You need to be authenticated in order to access this page</body></html>');
}
else if(auth) { //HAY DATOS DE AUTENTICACIÓN
 var separador = auth.split(' ');
 var buffer = new Buffer(separador[1], 'base64'); //Creamos un buffer en base64 y cogemos la segunda parte de lo separado
 var datosAuth = buffer.toString();
 var credenciales = datosAuth.split(':'); //Separamos por dos puntos para obtener username y contraseña
 var username = credenciales[0];
 var pass = credenciales[1];
 if(username=='xxxx'&&pass=='xxxx') { //COMPROBAR CREDENCIALES, Y SI SON CORRECTAS RENDERIZAR PAGINA DE ADMINISTRACIÓN
  res.statusCode = 200;  // OK
  LicenseKey.find({}, function (err, licenses) {
   if(licenses) {
    res.render('administration.ejs',{licencias: licenses});
   }
  });
 }
 else { // CREDENCIALES INCORRECTAS
  res.statusCode = 403; //Forbidden
  res.end('<html><body>You shall not pass</body></html>');
 }
}
```

Una vez superado el control de acceso, puede observarse dentro de la página web la información referente a las licencias, así como añadir nuevas dentro de la base de datos. Las licencias son introducidas en texto plano dentro de la web, y la lógica de aplicación se encargará de encriptarlas y almacenarlas, tal y como se muestra en el siguiente código:

```
var resultado = null;
var generateKey = function generateKey(){ //Generamos clave simétrica aleatoria
 var chars ="0123456789ABCDEFGHIJKLMNOPQRSTUVWXTZabcdefghiklmnopqrstuvwxyz*&-%/!?*+=()";
 var randomString = '';
 for (var i=0; i < 16; i++) {
```

```
    var rnum = Math.floor(Math.random() * chars.length);
    randomString += chars.substring(rnum,rnum+1);
   }
   return randomString;
};
var symmetricKey = generateKey();
var newLicense = new LicenseKey();
newLicense.cypherKey = symmetricKey;
var cypheredLicense = CryptoJS.AES.encrypt(req.body.license
    ,symmetricKey).toString(); // CIFRAR CON AES
newLicense.cypheredLicense = cypheredLicense;
newLicense.sold = false;
newLicense.queried = "None";
newLicense.registered = "None";
newLicense.save(function(err) { //GUARDAR
 var uri = 'https://'+req.headers.host.split(':')[0]+':'+'
    8443/administration';
 res.redirect(uri);
}
```

Perfil de usuario: Verificación de licencias y registro final de las mismas

Para acceder a las funcionalidades de verificación y registro de licencias se requiere que los usuarios inicien sesión previamente. La gestión en el proceso de registro y *login* es realizado por el módulo *PassportJS* incluido dentro del código, lo cual permite que la lógica sea muy sencilla de implementar:

```
 //Envio de datos de registro (POST)
app.post('/signup', passport.authenticate('register', {
 successRedirect: '/profile', // redirect to the secure
    profile section
 failureRedirect: '/signup', // redirect back to the signup
     page if there is an error
 failureFlash: true // allow flash messages
}));
/*****************************************************/
//Envio de datos de login (POST)
app.post('/signin', passport.authenticate('signin', {
 successRedirect: '/profile', // redirect to the secure
    profile section
 failureRedirect: '/signin', // redirect back to the signup
     page if there is an error
 failureFlash: true // allow flash messages
}));
```

Una vez realizado el inicio de sesión, es necesario gestionar desde la parte del servidor la información referente a las sesiones del usuario[3]. Para ello, se va a establecer la siguiente configuración en la dependencia de gestión de sesiones *express-session*:

[3]La información referente a sesiones se va a almacenar de manera local en la memoria del servidor en vez de almacenarla en la base de datos para agilizar el proceso

```
app.use(session(
{
 secret: 'jCNuQmgsub4H1WS0', //SECRETO
 resave: true, //SE GUARDA EL ESTADO DE LA SESIÓN AUNQUE NO SE REALICEN MODIFICACIONES DURANTE LA REQUEST
 saveUninitialized: true, //UNA SESIÓN NUEVA PERO NO MODIFICADA ES FORZADA PARA SER GUARDADA
 cookie: {
  secure: true //SOLO PERMITE HTTPS
  httpOnly: true //NO SE PUEDE ACCEDER POR MEDIO DE JAVASCRIPT
 }
}));
```

Una vez establecida la sesión del usuario, este puede acceder a las funcionalidades para verificar y/o registrar licencias. El código referente a estos métodos se presenta a continuación:

Comprobar validez de la licencia introducida

```
// Comprobamos validez licencia encriptada
var clave = req.body.license1;
var texto;
LicenseKey.findOne({'cypheredLicense': clave}, function (err, lic) { //BUSCAR LICENCIA INTRODUCIDA
 if (err) {
  res.json(err);
 }
 else if (lic) { //ENCONTRADA
  if (lic.registered == 'None') { // NO ESTA REGISTRADA
   if(lic.queried == 'None') { // NO SE HA CONSULTADO ANTES
    var dt = dateTime.create();
    var dtFormat = dt.format('Y-m-d H:M:S');
    lic.queried = req.user.name + ' at: '+dtFormat;
    lic.save(function (err) {
     if (err) {
      res.json(err);
     }
     else {
      texto = "ALL OK: The key you have introduced is not registered or queried";
     }
    });
   }
   else { // CONSULTADA PREVIAMENTE
    var datos = lic.queried;
    var partes = datos.split(" at: ");
    var fecha = partes[1];
    texto = "WARNING: The key you have entered has been queried by another user (DATE: "+fecha+"). "+
    "The order could be cancelled or the user has not exchange its final license. If you continue with the payment, do it at your own risk";
   }
  }
```

```
31  else { // REGISTRADA PREVIAMENTE
32   texto = "CAUTION: The key you have entered has been
        registered by another user. Please contact your
        distributor and request a valid license";
33  }
34 }
35 else { // NO EXISTE
36  texto = "CAUTION: The key you have entered is not
        registered in our database. Please contact your
        distributor and request a valid license";
37 }
38 });
39 req.flash("msg", texto);
40 res.locals.messages = req.flash();
41 res.locals.user = req.user;
42 res.render('profile');
```

Comprobar validez del pago y la licencia

```
1  // Comprobamos pago y damos licencia final al usuario
2  var licencia = req.body.license2;
3  var ticket = req.body.ticket;
4  var checkObj = JSON.stringify ( { //GENERAR OBJETO A ENVIAR
       A LA API
5   'licencia':  licencia,
6   'ticket': ticket
7  });
8  LicenseKey.findOne({'cypheredLicense': licencia}, function
       (err, lic) { //BUSCAMOS LA LICENCIA
9   if (err) {
10   res.json(err);
11  }
12  else if (lic) { // EXISTE LA LICENCIA
13   request (options, function (error, response, body) { //
        HACEMOS PETICIÓN A LA API
14    if(!error && response.statusCode == 200) { //RESPUESTA
         SATISFACTORIA
15     var claveFinal = CryptoJS.AES.decrypt(licencia,lic.
          cypherKey).toString(CryptoJS.enc.Utf8); //
          DESENCRIPTAR LICENCIA
16     var dt = dateTime.create();
17     var dtFormat = dt.format('Y-m-d H:M:S');
18     lic.registered = req.user.name + ' at: '+dtFormat;
19     lic.save(function (err) {
20      if (err) {
21       res.json(err);
22      }
23      else {
24       req.flash("licencia", claveFinal);
25       res.locals.messages = req.flash();
26       res.locals.user = req.user;
27       res.render('license');
28      }
29     });
30    }
```

```
31      else if (!error && response.statusCode ==404) { // EL
            TICKET NO EXISTE
32       req.flash("msg", "ERROR: We could not find any payment
             for this license. Make the payment or check if you
             have written the ticket reference correctly");
33       res.locals.messages = req.flash();
34       res.locals.user = req.user;
35       res.render('profile');
36      }
37      else if (!error && response.statusCode ==401) { // EL
            TICKET NO ESTA VINCULADO A LA LICENCIA
38       req.flash("msg", "ERROR: The ticket you have introduced
             is not related to the license.");
39       res.locals.messages = req.flash();
40       res.locals.user = req.user;
41       res.render('profile');
42      }
43     }).write(checkObj); //ESCRIBIR OBJETO JSON EN EL BODY DE
          LA PETICIÓN
44    }
45    else {
46     req.flash("msg","ERROR: We could not find a valid license
          . Check if you have introduced both fields properly.")
          ;
47     res.locals.messages = req.flash();
48     res.locals.user = req.user;
49     res.render('profile');
50    }
51   });
```

Compra de licencias por volumen

Para finalizar, se incluye el código referente a la generación del archivo JSON donde se almacenan todas las licencias encriptadas que adquiere el distribuidor:

```
1  var object = {};
2  var key = 'Licencias';
3  object[key] = []; //GENERAR ARRAY
4  LicenseKey.find({'sold': false}, function (err, licenses) {
       //BUSCAR LICENCIAS NO VENDIDAS
5   if (licenses) { //HAY SUFICIENTES LICENCIAS
6    licenses.forEach(function (item) { //INSERTAR CADA
         LICENCIA EN EL ARCHIVO JSON
7    item.sold = true; // Y MARCARLA COMO VENDIDA
8    item.save(function(err) {
9    });
10   object[key].push({
11    cypheredLicense: item.cypheredLicense
12   });
13  });
14  res.set('Content-Type', 'application/json');
15  res.json(object); //ENVIAR EL OBJETO JSON
16 }
17 }).limit(parseInt(cantidad)); //RECUPERAR DE LA DB
       UNICAMENTE EL N DE LICENCIAS A VENDER
```

Distribuidor

La lógica del distribuidor comparte muchas similitudes con el proveedor de licencias. Por tanto, en este subapartado solo se van a definir aquellos métodos y funcionalidades que son características de este elemento.

Administración de licencias

El método utilizado para proveer un sistema de autenticación que proteja la ruta es prácticamente idéntico al mostrado en el apartado del proveedor de licencias. La diferencia principal incluida en esta sección es la funcionalidad que permite al administrador subir las licencias en la base de datos a partir del fichero que ha generado el proveedor tras la compra por volumen. El código de esta funcionalidad se presenta a continuación:

```
upload(req, res, function(err){ // SE SUBE EL FICHERO
 if(err){
  res.json(err);
 }
 else if(req.file==undefined) { // NO SE SUBE FICHERO
  LicenseKey.find({}, function (err, licenses) {
   if(licenses) {
    req.flash("msg",'ERROR: Please upload a file before
        submitting');
    res.locals.messages = req.flash();
    res.render('administration.ejs', {
     licencias:licenses
    });
   }
  });
 }
 else { //SE SUBE, COMPROBAR MIME Y EXTENSION
  if(req.file.mimetype=='application/json' && req.file.
     originalname.split('.')[1]=='json') { // ES JSON
   var filePath = './uploads/'+req.file.filename; //
      RECUPERAR ARCHIVO SUBIDO
   var obj = JSON.parse(fs.readFileSync(filePath, 'utf8'));
       //LEERLO
   var saveLicencias = function(obj, callback) { //CICLO
      PARA GUARDAR LAS LICENCIAS
    async.forEach(obj.Licencias, function(data, callback) {
    var licencia = new LicenseKey();
    licencia.cypheredLicense = data.cypheredLicense;
    licencia.sold = false;
    licencia.ticket= 'None';
    licencia.queried = 'None';
    licencia.save(function(err) {
     if (err) {
      console.log(err);
     }
     else {
      console.log('saving Licencia');
       callback();
     }
    });
```

```
36      }, function(err) {
37        if (err) {
38          console.log(err);
39        }
40       });
41       res.redirect('administration'); // AL ACABAR DE GUARDAR
             , REDIGIR A LA PAGINA DE ADMIN
42      };
43      saveLicencias(obj); //EJECUTAR CICLO
44     }
45     else { // NO ES UN FICHERO JSON VÁLIDO
46      LicenseKey.find({}, function (err, licenses) {
47       if(licenses) {
48        req.flash("msg",'ERROR: Please upload a JSON FILE');
49        res.locals.messages = req.flash();
50        res.render('administration.ejs', {
51         licencias:licenses
52        });
53       }
54      });
55     }
56    }
57 });
```

Una vez se ha comprobado la validez del fichero y se suben las licencias, la página de administración será automáticamente recargada para mostrar en una tabla el conjunto de licencias actualizadas que se encuentran almacenadas en la base de datos.

Perfil de usuario: Compra de licencia y generación de *ticket*

Al igual que ocurría en la parte del proveedor de licencias, resulta necesario que el cliente inicie sesión dentro de la plataforma para poder acceder a las rutas protegidas. La gestión del registro de usuarios, *login* y control de sesiones se va a realizar utilizando *PassportJS*, así como los métodos que se han explicado con anterioridad en el apartado del proveedor.

Una vez autenticado, el usuario puede acceder a la página para solicitar la compra de una licencia. Una vez introducidos los datos, se procede al pago y a la generación del *ticket*. Para ello, se utiliza el siguiente código:

```
1  if(blocked) { // SI NO HAY STOCK BLOQUEAMOS LA COMPRA
2   req.flash("msg", "ERROR: There is not stock. Wait for a
       refill.");
3   req.flash("license","Not enough stock");
4   res.locals.licencias = req.flash();
5   res.render('purchase.ejs');
6  }
7  else { //HAY STOCK Y SE PUEDE PAGAR
8   request(options, function (error, response, body) { //
       PETICIÓN A LA API
9    if (!error && response.statusCode == 200) { // TODO
       CORRECTO
```

```
10      var obj =JSON.parse(licenciaJSON); //OBTENEMOS EL TICKET
            GENERADO
11      LicenseKey.findOne({'cypheredLicense': obj.licencia},
            function (err, license) {
12       if (license) {
13        license.ticket = CryptoJS.AES.encrypt(response.body,
              symmetricKey).toString(); // AÑADIMOS EL TICKET
              ENCRIPTADO
14
15        license.save(function(err) { //GUARDAMOS EL TICKET
16         req.flash("licencia", obj.licencia);
17         req.flash("ticket", response.body);
18         res.locals.messages = req.flash();
19         res.render('ticket.ejs');
20        });
21       }
22      });
23     }
24    }).write(licenciaJSON); //ENVIAMOS LA LICENCIA QUE SE VA A
          PAGAR
25  }
```

Por otro lado, la configuración de la petición realizada al gestor de pago tiene la siguiente forma:

```
1  var options = {
2   url: 'https://localhost:8445/doPayment', //DIRECCIÓN
3   method: 'post', //METODO POST
4   headers: {
5    'authorization': 'G2A:LrDXJ5erVtQVS6kFbazWqUHa482CfG2e',
         //DATOS DE IDENTIDAD
6    'Content-Type': 'application/json' // DATOS EN FORMATO
         JSON
7   },
8   rejectUnauthorized:false // PUESTO QUE USAMOS CERTIFICADOS
        AUTOFIRMADOS, SI NO SE DESACTIVA ESTA PROTECCIÓN NO SE
        PUEDE UTILIZAR EL METODO. DE NORMAL, ESTA ACTIVADO
9  };
```

Gestor de pagos

El gestor de pagos actúa de manera similar a una *API*. La lógica de aplicación dispone de dos métodos, uno para realizar los pagos de las licencias y otro para verificar el estado del pago de las licencias. Los métodos se encuentran debidamente protegidos, de forma que unicamente las empresas puedan acceder a los mismos. La autenticación se basa en el análisis de las cabeceras para comprobar si se ha enviado información referente a la identidad del agente que realiza la petición. Esta información debe ser enviada siguiendo la siguiente estructura:

nombre:código

Si se considera como ejemplo una petición enviada desde el distribuidor al gestor de pagos, dentro de la cabecera se puede observar la siguiente información:

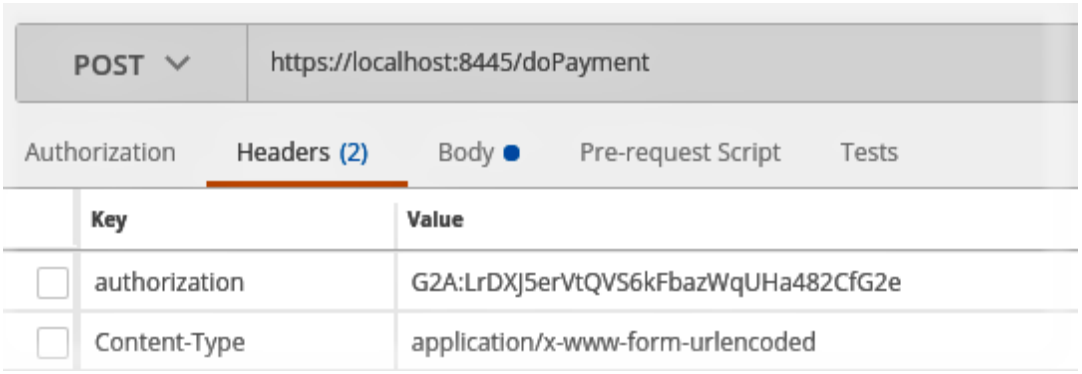


FIGURA II.1: Identificación de empresas enviada en las cabeceras

Antes de realizar cualquier operación, el gestor de pagos ejecuta el siguiente código para verificar que la identidad del agente que realiza la consulta es correcta:

```
1  var auth = req.headers['authorization'];
2  if(!auth) { //NO SE ENVÍAN CABECERAS
3   res.statusCode = 401;
4   res.send('ERROR: You need to be authenticated in order to
       use this API');
5  }
6  else { //HAY INFORMACIÓN DE IDENTIDAD EN LAS CABECERAS
7   var credenciales = auth.split(':');
8   var name = credenciales [0];
9   var code = credenciales [1];
10  empresa.findOne({'name': name, 'code': code}, function (
       err, ent) { //BUSCAMOS SI EXISTE EL REGISTRO
11   if(err) {
12    res.json(err); //ERROR EN LA CONEXIÓN
13   }
14   else if(ent) { // IDENTIFICACIÓN CORRECTA
15     //EJECUTAR MÉTODO
16   }
17   else { //LOS DATOS SON INCORRECTOS
18    res.statusCode = 403;
19    res.send('FORBIDDEN');
20   }
21  });
22 }
```

Una vez verificada la identidad del agente, se puede proceder con la ejecución de las funcionalidades referentes a la realización de pagos y la generación de sus correspondientes *tickets*, así como la validación de los mismos a la hora de entregar la licencia final al usuario:

Pago y generación del *ticket*

```
var p = new pago();
var fecha = new Date().getTime().toString();
var aleatorio = Math.round(Math.random()*1000).toString();
    //NUMERO ALEATORIO
p.chain = fecha+'.'+aleatorio; //CADENA FORMADA POR LA
    FECHA EN MS Y EL NUMERO ALEATORIO

var cadena = req.body.licencia+'.'+p.chain; //AÑADIMOS LA
    LICENCIA A LA CADENA
p.ticket = crypto.createHash('md5').update(cadena).digest('
    hex'); //GENERAR HASH

p.save(function (err) { //GUARDAR TICKET EN LA DB
 if(err) {
  res.json(err);
 }
 else {
  res.send(p.ticket); //ENVIAR TICKET AL USUARIO
 }
});
```

Comprobación del pago y validación del *ticket*

```
pago.findOne({'ticket':ticket}, function (err,pg) { //
    BUSCAMOS EL TICKET DENTRO DE LA DB
 if(err) {
  res.json(err);
 }
 else if (pg) { // SE ENCUENTRA EL TICKET
  var cadena = licencia+'.'+pg.chain; //GENERAMOS EL CÓDIGO
      CON LA LICENCIA INTRODUCIDA Y LA CADENA
  var hash = crypto.createHash('md5').update(cadena).digest
     ('hex'); //GENERAMOS RESUMEN
  if(hash == pg.ticket) { // SI SON IGUALES, OK
   res.statusCode = 200;
   res.send('OK');
  }
  else { // SI NO SON IGUALES NO ES VÁLIDO
   res.statusCode = 401;
   res.send('NOT MATCHED');
  }
 }
 else { //EL TICKET INTRODUCIDO NO EXISTE
  res.statusCode = 404;
  res.send('NOT FOUND');
 }
});
```